Audio Architecture in Raspberry Pi

This paper presents a comprehensive analysis of the audio architecture employed by the Raspberry Pi for audio playback and capture using speakers and microphones. By examining the low-level audio framework and referencing code examples from the Raspberry Pi GitHub repository, we delve into the implementation of the Advanced Linux Sound Architecture (ALSA) software stack. This investigation provides valuable insights into the underlying mechanisms that enable the Raspberry Pi to effectively handle audio tasks.

ALSA, Linux Kernel, Embedded Audio, Raspberry Pi

Introduction

The Advanced Linux Sound Architecture (ALSA) is a fundamental component of the Linux operating system, serving as the audio engine for the Raspberry Pi. ALSA provides a standardized interface for applications to interact with various sound hardware devices, including microphones, speakers, and sound cards. This enables the Raspberry Pi to handle a wide range of audio tasks, such as:

- Multimedia playback: Playing music, watching videos, and streaming audio.
- Gaming: Providing sound effects and music for games.
- Voice communication: Enabling voice calls and online meetings.
- Audio recording: Capturing audio from external sources.
- Audio processing: Applying effects like equalization, compression, and reverb.

By leveraging ALSA, the Raspberry Pi offers versatile audio capabilities, making it a popular choice for sound and multimedia projects.

ALSA Components

ALSA Components

ALSA Components

To convert digital audio data (e.g., MP3 or WAV files) into analog audio for playback through speakers, several components within ALSA work together:

- **Codec Driver:** This driver interfaces with the hardware codec (e.g., from Wolfson or Maxim) responsible for converting digital audio signals to analog and vice versa.
- **Platform Driver:** Specific to the Raspberry Pi board, this driver handles communication between the CPU and the audio hardware, typically using the I2S

(Inter-Integrated Circuit Sound) interface to transfer audio data to and from the codec.

- **Machine Driver:** Tailored to the Raspberry Pi, this driver orchestrates interactions between the codec, platform driver, and other board-specific components. It manages tasks such as clock control, GPIO configuration, and enabling/disabling audio features to ensure proper audio functionality.

The machine driver on a Raspberry Pi contains the following code, which connects all the components:

```
SND_SOC_DAILINK_DEFS(rpi_proto,
    DAILINK_COMP_ARRAY(COMP_CPU("bcm2708-i2s.0")),
    DAILINK_COMP_ARRAY(COMP_CODEC("wm8731.1-001a", "wm8731-hifi")),
    DAILINK_COMP_ARRAY(COMP_PLATFORM("bcm2708-i2s.0")));
```

- **COMP_CPU:** This macro provides the name of the CPU driver (platform driver), which is "bcm2708-i2s". It is a Broadcom driver for transmitting audio data over the I2S bus to a codec capable of processing input and converting it to analog output using a DAC (digital-to-analog converter).
- **COMP_CODEC:** This macro specifies the codec driver from Wolfson, which contains a DAC that performs the necessary analog conversion for audio playback.
- **COMP_PLATFORM:** Similar to the CPU driver, this macro is included to maintain proper functionality of the ALSA framework.

Components Responsibilities

The machine driver is responsible for populating all fields in the `snd_soc_dai_link` structure and invoking the `devm_snd_soc_register_card()` function to register the sound card, referred to as `snd_rpi_proto`. Additionally, it manages clock setup and provides necessary configuration information to the CPU/platform and codec drivers. As detailed in , the `snd_rpi_proto_hw_params` function calls `snd_soc_dai_set_bclk_ratio` and `snd_soc_dai_set_sysclk` to establish the I2S bit clock and crystal (XTAL) frequency for the codec driver.

The `bcm2708-i2s.c` driver registers itself as a platform driver by calling `devm_snd_soc_register_component`. It then invokes `devm_snd_dmaengine_pcm_register` for DMA operations. When playback starts, DMA is triggered, and the `bcm2835_i2s_dai_ops` operations enable the clock for DMA.

The DMA driver configures the register space of the DMA device, initiated by the ALSA core's DMA engine. The DMA engine creates a PCM platform driver that translates user space configuration parameters (e.g., sample rate, bit width) into DMA hardware configuration parameters, keeping the DMA engine active as long as audio data is being played. I2S hardware is typically augmented with FIFO, configured by the DMA hardware to generate interrupts when a portion of FIFO is filled, facilitating efficient data transfer.

The codec driver is responsible for converting digital audio to analog via a DAC. Configuration settings such as clock, bit width, and sample rate are managed within the codec driver. The `wm8731.c` file in the contains several key functions:

- `wm8731_hw_params`: Sets the bit depth of the codec.
- `wm8731_set_dai_fmt`: Configures the interface format and clock polarity.

In summary, the codec driver is responsible for setting the I2S bus characteristics and its own hardware settings to receive audio from the CPU and convert it to an analog format for output to speakers.

The ALSA core exposes a device node to user space, allowing applications to play or capture audio. All components—platform/CPU, codec, and machine driver—provide common function calls as shown below:

```
/* SoC audio ops */
struct snd_soc_ops {
    int (*startup)(struct snd_pcm_substream *);
    void (*shutdown)(struct snd_pcm_substream *);
    int (*hw_params)(struct snd_pcm_substream *, struct
snd_pcm_hw_params *);
    int (*hw_free)(struct snd_pcm_substream *);
    int (*prepare)(struct snd_pcm_substream *);
    int (*trigger)(struct snd_pcm_substream *, int);
};
```

These operations are triggered by the ALSA core when playback or capture begins from user space.

Conclusion

To enable audio playback or capture on the Raspberry Pi, the ALSA framework interacts with the codec, platform/CPU, and machine driver to present a sound card to user space. User space utilizes the ALSA library to read and write audio data to the sound card, facilitating audio playback and capture.

00 [Online]. Available: https://en.wikipedia.org/wiki/I%C2%B2S [Online]. Available: https://github.com/thenaran/linux-rpi/blob/master/sound/soc/bcm/bcm2708-i2s.c [Online]. Available: https://github.com/raspberrypi/linux/blob/rpi-6.6.y/sound/soc/codecs/wm8731.c [Online]. Available: https://github.com/raspberrypi/linux/blob/rpi-6.6.y/sound/soc/bcm/rpi-proto.c [Online]. Available: https://en.wikipedia.org/wiki/Pulse-code_modulation [Online]. Available: https://elixir.bootlin.com/linux/latest/source/sound/soc/raspberrypi/rpi-dma.c