



Audio codecs hardware components



Contents

1. Audio codecs hardware components	3
2. ALSA overview	9
3. Category:ST boards	22



A quality version of this page, approved on 2 June 2020, was based off this revision.

Contents

1 Article purpose	4
2 Software frameworks	5
3 Wolfson wm8994	6
3.1 Linux driver	6
4 Cirrus cs42151	7
4.1 Linux driver	7
5 Silab sil9022	8
5.1 Linux driver	8
6 ST MP34DT01-M	9
6.1 Linux driver	9



1 Article purpose

The purpose of this article is to:

- list the audio codec hardware components that might be integrated in the different STM32MPU boards
- link these components to the corresponding software framework(s)
- point to the datasheet(s) of these components
- explain, when necessary, how to configure these components.



2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Audio	Wolfson wm8994		ALSA framework	DAC & ADC audio codec
Audio	Cirrus cs42l51		ALSA framework	DAC & ADC audio codec
Audio	Silab sil9022		ALSA framework	HDMI transmitter
Audio	ST MP34DT01- M		ALSA framework	Digital microphone



3 Wolfson wm8994

The WM8994 is a highly integrated ultra-low power hi-fi CODEC rich in multimedia features.

For details and the datasheet please contact the WM8994 codec provider.

3.1 Linux driver

Bindings: [Documentation/devicetree/bindings/sound/wm8994.txt](#)

Sources:

- [drivers/mfd/wm8994-core.c](#)
- [sound/soc/codecs/wm8994.c](#)
- [sound/soc/codecs/wm_hubs.c](#)



4 Cirrus cs42l51

The CS42L51 is a highly integrated, 24-bit, low-power stereo CODEC.

For details and the datasheet please contact the CS42L51 codec provider.

4.1 Linux driver

Bindings: [Documentation/devicetree/bindings/sound/cs42l51.txt](#)

Sources: [sound/soc/codecs/cs42l51.c](#)



5 Silab sil9022

The Sil9022A HDMI transmitter supports the HDMI[®] Specification on a wide range of mobile products.
For details and the datasheet please contact the Sil9022A HDMI transmitter provider.

5.1 Linux driver

Bindings: `Documentation/devicetree/bindings/display/bridge/sii902x.txt`

Sources: `drivers/gpu/drm/bridge/sii902x.c`



6 ST MP34DT01-M

The MP34DT01-M is an ultra-compact, low-power, omnidirectional, digital MEMS microphone.

Datasheet: <https://www.st.com/resource/en/datasheet/mp34dt01-m.pdf>

6.1 Linux driver

Bindings: Documentation/devicetree/bindings/sound/dmic.txt

Sources: sound/soc/codecs/dmic.c

Cortex[®]

Open Portable Trusted Execution Environment

Linux[®] is a registered trademark of Linus Torvalds.

Digital-to-analog converter (Electronic circuit that converts a binary number into a continuously varying value.)

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

High-Definition Multimedia Interface (HDMI standard)

Stable: 03.01.2021 - 15:38 / Revision: 04.01.2021 - 17:18

A quality version of this page, approved on 5 January 2021, was based off this revision.

This article gives information about the Advanced Linux Sound Architecture (ALSA), which provides audio functionality to the Linux[®] operating system.

Contents

1 Purpose	11
2 System overview	12
2.1 Component descriptions	12
2.2 API descriptions	14
3 Configuration	15
4 How to use	16
4.1 Playback	16
4.2 Record	16
4.3 Controls	16
4.4 IEC controls	17
5 How to trace and debug the framework	18
5.1 How to monitor	18
5.1.1 Procfs filesystem	18
5.1.2 Debugfs filesystem	18
5.2 How to trace	18
5.2.1 Dynamic traces	18
5.2.2 Tracing filesystem	19
5.2.2.1 Activate DAPM traces	19



5.2.2.2 Activate PCM hardware parameter traces	19
5.2.2.3 Activate PCM buffer state traces (PCM ring buffer overrun/underrun debugging)	19
5.3 How to debug	20
6 Source code location	21
6.1 User space	21
6.2 Kernel space	21
7 References	22

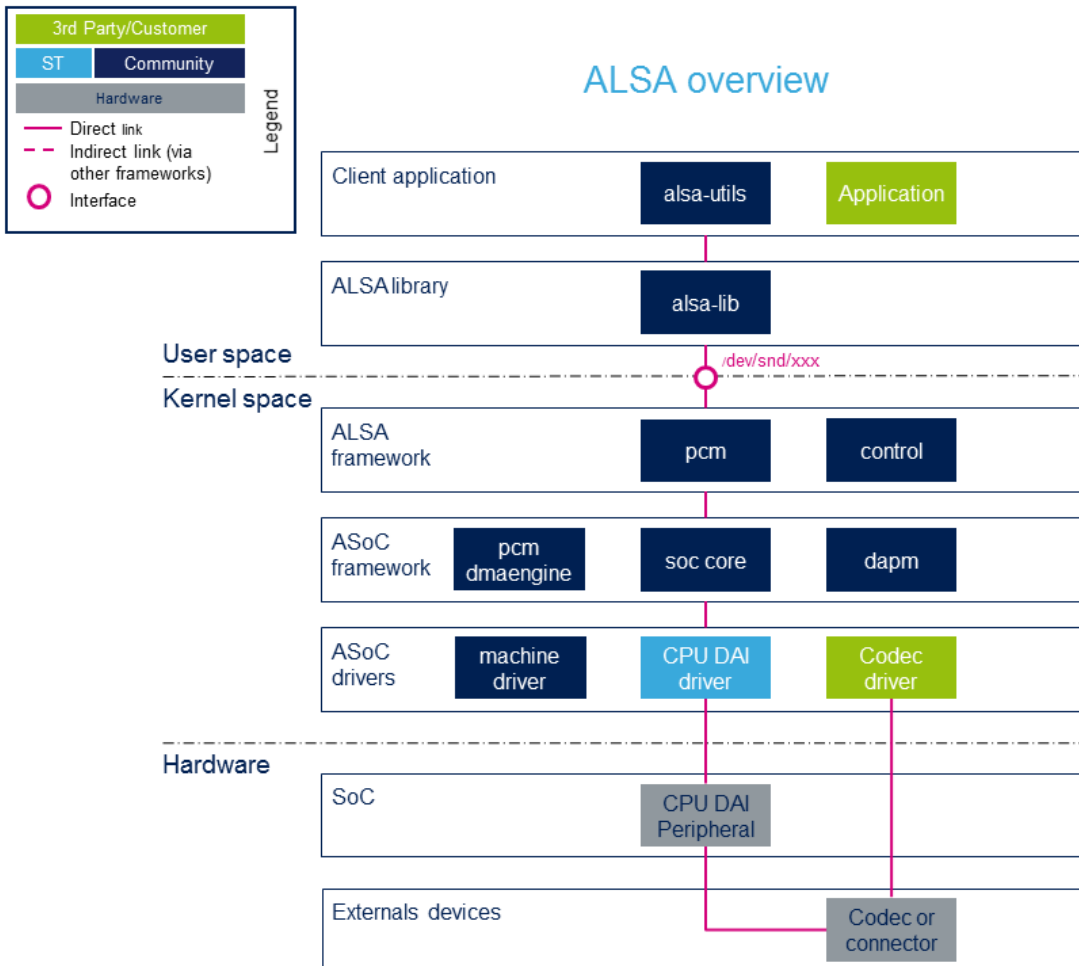


1 Purpose

The purpose of this article is to introduce the ALSA framework.

The ALSA framework provides comprehensive audio functionality for Linux which includes recording and playing of audio streams, in either analog or digital formats together with routing and mixing capabilities. ALSA also supports audio middleware such as PulseAudio, Gstreamer or Android.

2 System overview



2.1 Component descriptions

- **alsa-utils** (User space)

The ALSA utility package, provided by the Linux community, contains the command line utilities for the ALSA project (aplay, arecord, amixer, alsamixer ...). These tools are useful for controlling soundcards. They also provide an example of ALSA API use, for application implementation.

- **alsa-lib** (User space)

The ALSA Library package contains the ALSA library used by programs (for instance alsa-utils programs) requiring an access to the ALSA sound interface. The ALSA library provides a level of abstraction, such as the PCM and control abstractions, over the audio devices provided by the kernel modules.

- **ALSA framework** (Kernel space)



The ALSA core provides an API to implement audio drivers and PCM/control interfaces to expose audio devices on the userland. The PCM interface handles the data flow and control. The interface manages controls exported by the ALSA driver (audio path, volumes...).

- **ASoC framework (ALSA System On Chip)** (Kernel space)

The aim of the ALSA System on Chip (ASoC) layer^[1] is to improve ALSA support for embedded system-on-chip processors and audio codecs. The ASoC framework provides a DMA engine which interfaces with DMA framework to handle the transfer of audio samples. ASoC also supports the dynamic power management of audio paths through the DAPM driver. ASoC acts as an ALSA driver, which splits an embedded audio system into three types of platform independent drivers: the CPU DAI, the codec and the machine drivers.

- **ASoC drivers** (Kernel space)

ASoC drivers allow the implementation of hardware dependent code for ASoC driver classes:

- **Codec drivers:**

These drivers are the drivers for the backend audio components. (see Codec peripherals below)

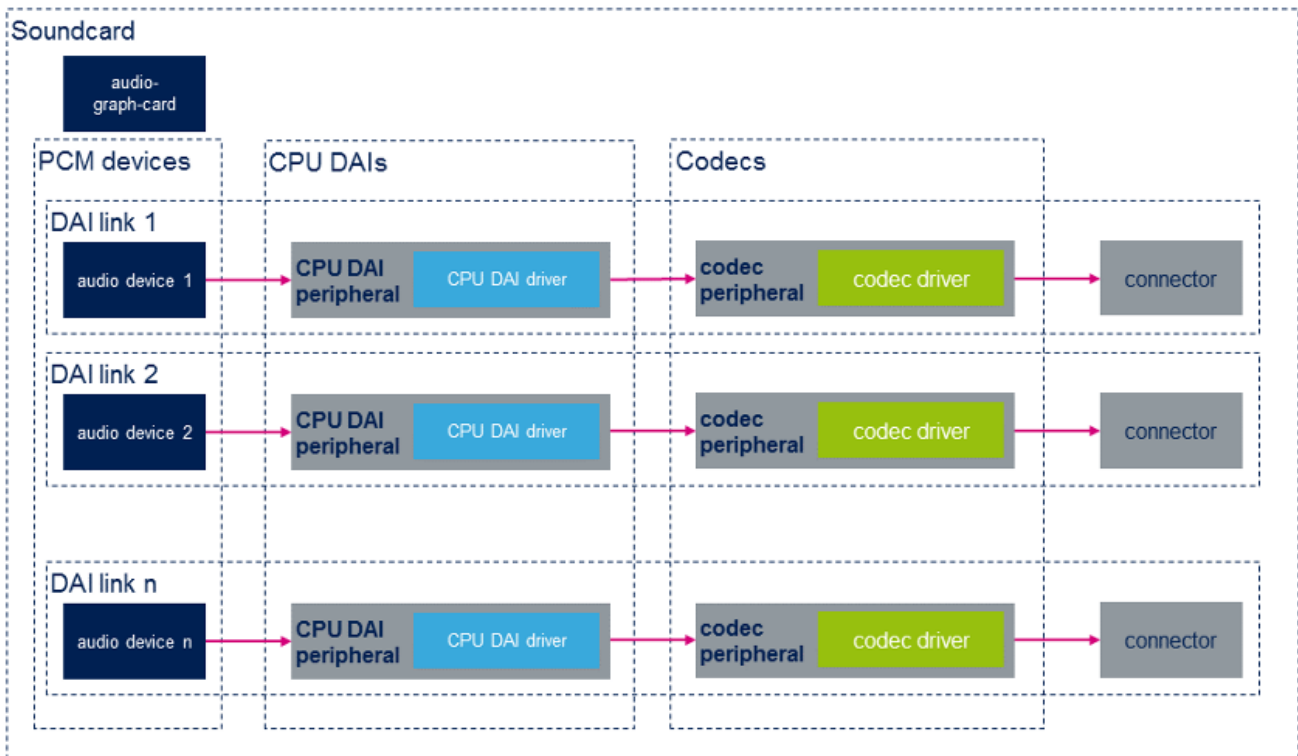
- **CPU DAI drivers:**

There is a specific CPU DAI driver for each STM32 audio peripheral (see CPU DAI peripherals below). Each CPU DAI supports at least one of the following protocols: I2S, PCM, or S/PDIF.

- **Machine drivers:**

The machine drivers describe and bind the CPU DAIs and codec drivers together to create the DAI links and ALSA soundcard. The ASoC framework provides a machine driver implementing a generic soundcard called "audio-graph-card"^{[2][3]}. This generic machine driver is used for STM32 MPUs soundcards.

The schematic below, illustrates the general layout of an ASoC soundcard. Refer to [soundcard configuration](#) to see examples of soundcards implementation for STM32 MPUs boards.



- **CPU DAI peripherals (Hardware)**

The ST microprocessor peripheral provides the CPU audio interface. The audio internal peripheral list can be found in [Audio peripherals](#) section.

- **Codec peripherals (Hardware)**

The codec peripherals are the external (none CPU) hardware audio I/O devices (i.e. audio codec IC, digital microphone, amplifier, simple IO connector ...).

2.2 API descriptions

- **User space interface:**

The ALSA library reference ^[4] documents the userland API library.

- **Kernel driver interface:**

The ALSA kernel documentation ^[5] documents the ASOC and ALSA driver APIs.



3 Configuration

- **Kernel Configuration**

The ALSA/ASoC and the audio graph card must be enabled in the kernel configuration, as shown below, to enable the sound support. On top of this, the user has to activate the CPU and Codec drivers according to the chosen hardware. The user can use Linux [Menuconfig tool](#) to select the required drivers:

```
[*] Device Drivers
  [*] Sound card support
    [*] Advanced Linux Sound Architecture
      [*] ALSA for SoC audio support
        STMicroelectronics STM32 SOC audio support
          [ ] STM32 SAI interface (Serial Audio Interface) support
          [ ] STM32 I2S interface (SPI/I2S block) support
          [ ] STM32 S/PDIF receiver (SPDIFRX) support
        CODEC drivers
          [ ] ...
      [*] ASoC Audio Graph sound card support
```

- **Device tree configuration**

The audio sub-system is configured through the soundcard configuration in the [device tree](#). The [soundcard configuration](#) article describes the soundcards available for STM32MPUs on various board. This article details how to configure the [audio peripherals](#) used to implement the soundcards.



4 How to use

The `alsa-utils` package provides a set of utilities to manage audio devices in the Linux kernel: `aplay`, `arecord`, `amixer`, `iecset` and `alsactl`. An overview of these utilities is given below:

4.1 Playback

- List playback devices

```
Board $> aplay -l
```

- Play a wav file on card [X] device [Y]

```
Board $> aplay -D hw:[X],[Y] <filename.wav>
```

- Play a wav file or a generated signal on card [X] device [Y]

```
Board $> speaker-test -D hw:[X],[Y]
```

Refer to [How to play audio](#) article, to find examples of playback use cases on STM32MPU boards.

4.2 Record

- List record devices

```
Board $> arecord -l
```

- Capture audio from card [X] device [Y]

```
Board $> arecord -D hw:[X],[Y] -f dat <filename.wav>
```

Refer to [How to record audio](#) article, to find examples of record use cases for the STM32MPU boards.

4.3 Controls

- List card [X] controls

```
Board $> amixer -c [X] controls
```

- Set card [X] controls [Y] to value [Z]



```
Board $> amixer -c [X] cset name='[Y]' '[Z]'
```

- Store soundcard [X] controls state

```
Board $> alsactl store [X]
```

- Restore soundcard [X] controls state

```
Board $> alsactl restore [X]
```

Refer to [Soundcard configuration](#) article to find examples of control configuration for the STM32MPU boards.

4.4 IEC controls

- List iec958 parameters

```
Board $> iecset -h
```

- Set card [X] iec958 parameter [Y] to value [Z]

```
Board $> iecset -c [X] cset [Y] [Z]
```

- Dump card [X] iec958 value

```
Board $> iecset -c [X] -x
```



5 How to trace and debug the framework

This chapter introduces tools useful for debugging and monitoring the audio framework and drivers. It comes as an extension to the [Linux_tracing,_monitoring_and_debugging](#) article.

5.1 How to monitor

This section introduces ALSA framework monitoring methods. Refer to [Linux monitoring tools](#) articles for further information.

5.1.1 Procfs filesystem

The ALSA **asound** directory^[6] in **procfs** file system, provides a lot of information on sound cards. The PCM proc files, provides useful PCM substream debugging information, such as hardware/software parameters, stream status and buffer information.

Examples:

- List PCM audio devices:

```
Board $> cat /proc/asound/pcm
```

- Get hardware parameters of a PCM audio device (device "0" of card "0" here):

```
Board $> cat /proc/asound/card0/pcm0p/sub0/hw_params
```

5.1.2 Debugfs filesystem

The **asoc** directory in **debugfs** file system provides information on sound card components.

Examples:

- List DAIs

```
Board $> cat /sys/kernel/debug/asoc/dais
```

- List DAPMs of "xxx.audio-controller" CPU DAI of "STM32MP1-EV" soundcard

```
Board $> ls /sys/kernel/debug/asoc/STM32MP1-EV/xxx.audio-controller/dapm
```

5.2 How to trace

This section introduces tracing methods for ALSA framework. Refer to [Linux_tracing_tools](#) articles to go further.

5.2.1 Dynamic traces

ALSA framework and driver debug traces can be added to the kernel logs by using the [dynamic debug](#) mechanism.

- Example: Activate dynamic trace for SAI Linux driver, and print the traces to the console:



```
Board $> echo -n 'file stm32_sai.c +p; file stm32_sai_sub.c +p' > /sys/kernel/debug
/dynamic_debug/control;
Board $> dmesg -n8;
```

5.2.2 Tracing filesystem

The Linux kernel offers a `tracefs` filesystem, provided with Linux kernel tracing framework. ALSA and ASoC have their own tracepoints in this tracing filesystem:

- `asoc` entry for ASoC, provides the DAPM, jack and bias level tracepoints.^[7]
- `snd_pcm` entry for ALSA, provides the PCM buffers and PCM hardware parameters tracepoints^{[7][8]}.

5.2.2.1 Activate DAPM traces

Prerequisite: the `CONFIG_FUNCTION_TRACER` configuration must first be enabled in the Linux kernel configuration

- Enable trace^[7]

```
Board $> echo '1' > /sys/kernel/debug/tracing/events/asoc/enable
```

- Check log:

```
Board $> cat /sys/kernel/debug/tracing/trace
```

5.2.2.2 Activate PCM hardware parameter traces

Prerequisite: the `CONFIG_FUNCTION_TRACER` and `CONFIG_SND_DEBUG` configurations must first be enabled in the Linux kernel configuration

- Enable trace^[7]

```
Board $> echo '1' > /sys/kernel/debug/tracing/events/snd_pcm/enable
```

- Check log:

```
Board $> cat /sys/kernel/debug/tracing/trace
```

5.2.2.3 Activate PCM buffer state traces (PCM ring buffer overrun/underrun debugging)

Prerequisite: the `CONFIG_FUNCTION_TRACER`, `CONFIG_SND_DEBUG`, `CONFIG_SND_DEBUG_VERBOSE` and `SND_PCM_XRUN_DEBUG` configurations must first be enabled in the Linux kernel configuration

- Set XRUN trace verbosity^[9]

```
# Enable basic debugging and stack dump
Board $> echo 3 > /proc/asound/card0/pcm0p/xrun_debug
```

- Enable trace^[7]



```
Board $> echo '1' > /sys/kernel/debug/tracing/events/snd_pcm/enable
```

- Check log:

```
Board $> cat /sys/kernel/debug/tracing/trace
```

5.3 How to debug

Refer to the [Linux debugging tools](#) articles.



6 Source code location

6.1 User space

- alsa-lib sources
- alsa-utils sources

6.2 Kernel space

- ALSA core
- ASoC core
- ASoC STM32 drivers
- ASoC codecs drivers



7 References

- ASoC layer documentation
- Documentation/devicetree/bindings/sound/audio-graph-card.txt
- Documentation/devicetree/bindings/graph.txt
- ALSA library API
- ALSA and ASoC driver API documentation
- ALSA proc files
- 7.07.17.27.37.4 Documentation/trace/tracepoint-analysis.rst
- ALSA tracepoints
- XRUN Debug

Linux[®] is a registered trademark of Linus Torvalds.

Advanced Linux sound architecture

Application programming interface

ALSA System on Chip

Direct Memory Access

Dynamic Audio Power Management

Central processing unit

Digital Audio Interface

Integrated Interchip Sound

Sony/Philips Digital Interface Format (Protocol (IEC-60958))

input/output

Serial Audio Interface (Mechanism used to transfer non-buffered audio data between processors and/or audio converters.)

Serial Peripheral Interface

Stable: 17.06.2020 - 15:27 / Revision: 16.01.2020 - 13:36

A quality version of this page, approved on 17 June 2020, was based off this revision.

This category groups together all articles related to any STMicroelectronics board.



Pages in category "ST boards"

The following 8 pages are in this category, out of 8 total.

- MB1230
- MB1262
- MB1263
- MB1272
- MB1379
- MB1407
- STM32MP157x-DKx - hardware description
- STM32MP157x-EV1 - hardware description