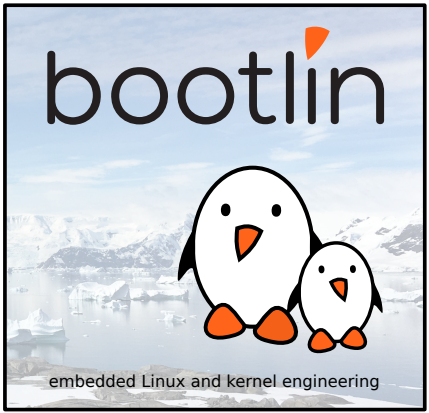
ASoC: Supporting Audio on an Embedded Board

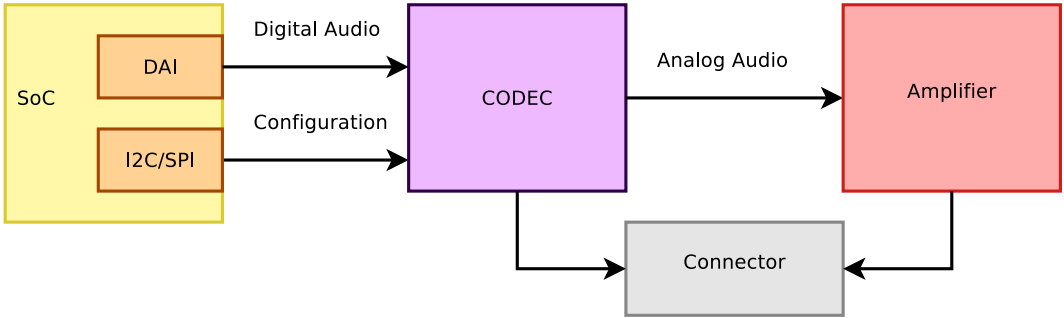
Alexandre Belloni
Bootlin
alexandre.belloni@bootlin.com

Alexandre Belloni

- Embedded Linux engineer at Bootlin
  - Embedded Linux **expertise**
  - **Development**, consulting and training
  - Bootloader, Linux kernel, Yocto Project, Buildroot
  - Complete Linux BSP development
  - Hardware support in bootloader/Linux
  - Strong open-source focus: upstreaming and contributions
  - Freely available training materials
- Open-source contributor
  - Maintainer for the Linux kernel **RTC subsystem**
  - Co-Maintainer of **kernel support for Microchip (ARM and MIPS) processors**



Anatomy

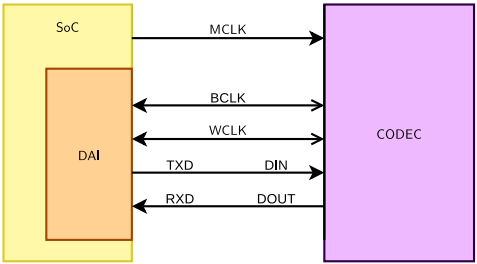

SoC
DAI
I2C/SPI
Digital Audio
Configuration
CODEC
Analog Audio
Amplifier
Connector

Anatomy

- codec configuration usually happens on a simple serial bus, I2C or SPI.
- SoC DAI: the SoC Digital Audio Interface.
  - sometimes called synchronous serial interface
  - provides audio data to the codec
  - formats are usually AC97, I2S, PCM (TDM, network mode), DSP A/B
  - Examples: Atmel SSC, NXP SSI, TI McASP.
  - Some SoCs have a separate SPDIF controller
- Amplifier is optional

Some SoCs (Allwinner A33, Atmel SAMA5D2) have the codec and the amplifier on the SoC itself.

Digital formats - signals



Digital formats - DAI

- The Digital Audio Interface is a serial interface
- It uses two clocks: the bit clock and the frame clock.
  - The bit clock is usually referred to as BCK or BCLK
  - The frame clock is often called FCLK/FSCK/FSCLK, LRCK/LRCLK (Left Right clock) or WCLK (word clock). Its rate is the sample rate also called Fs.
  - The relationship between BCK and FSCK is: $bck = fsck * Nchannels * BitDepth$
- It also has one or multiple data lines.

MCLK

- MCLK is the codec clock. It is sometimes referred as the system clock. The IC needs it to be working.
- Some codecs will also require it to be able to use the control interface.
- Can be provided by the SoC when it has suitable clocks or a crystal.
- Some codecs are able to use BCLK or LRCLK as their clock, making MCLK optional.
- usually the codecs will expect MCLK to be a multiple of BCLK. Usually specified as a multiple of Fs.

Clocks: master/slave

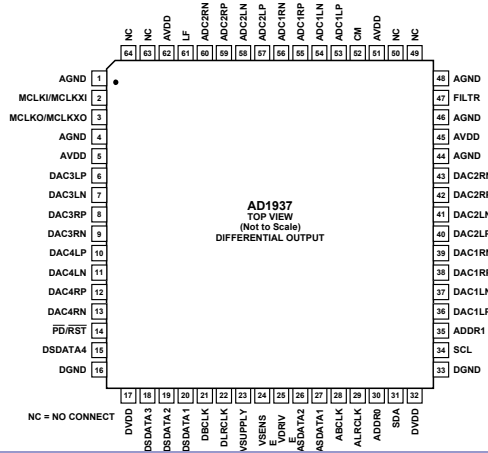
- One of the DAI is responsible to generate the bit clock, it is the bit clock master.
- One of the DAI is responsible to generate the frame clock, it is the frame master.
- Some have a great set of PLLs and dividers, allowing to get a precise BCLK from many different MCLK rates.
- Quite often, it is better to use the codec as master. However, some SoCs have specialized audio PLLs.

Digital formats - Data

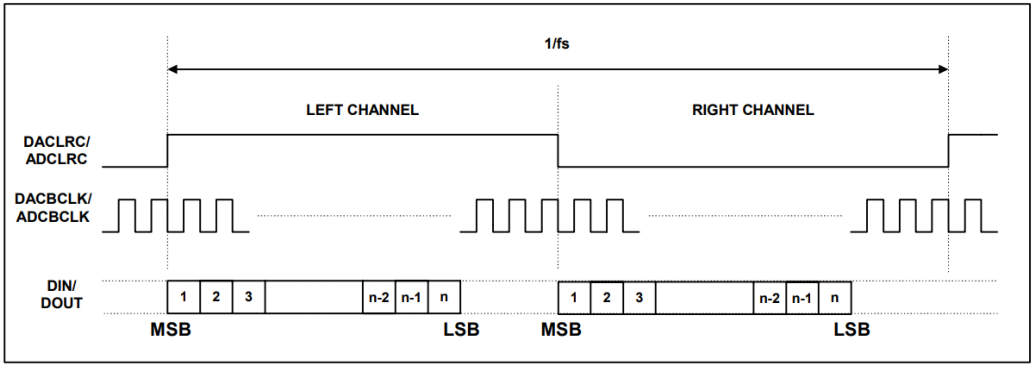
- Codecs may have multiple data in or data out lines, one line per channel pair.
- Codecs may also have multiple DAI, one full interface for data in and one for data out.

e.g. AD1937 has:

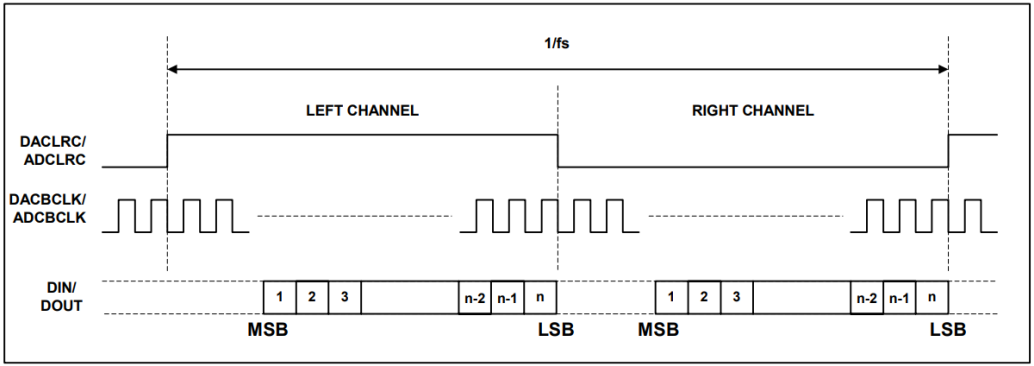
- 8 DACs in 4 pairs, 4 ADCs in 2 pairs
- clocks for data-in: DBCLK, DLRCLK
- 4 data-in lines (DSDATA[1-4])
- clocks for data-out: ABCLK, ALRCLK
- 2 data-out lines (ASDATA[1-4])



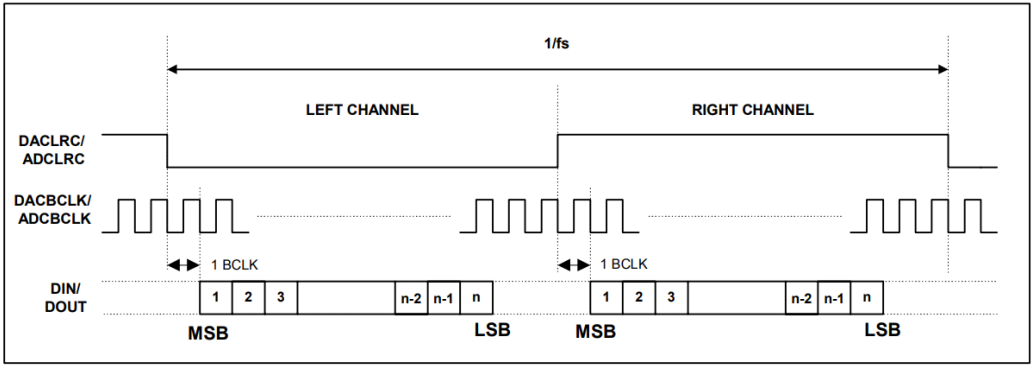
Digital formats - Left Justified



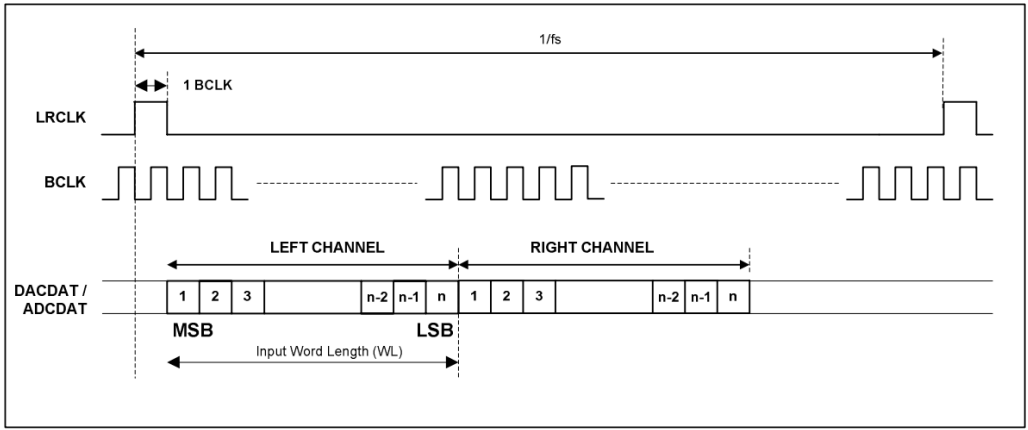
Digital formats - Right Justified



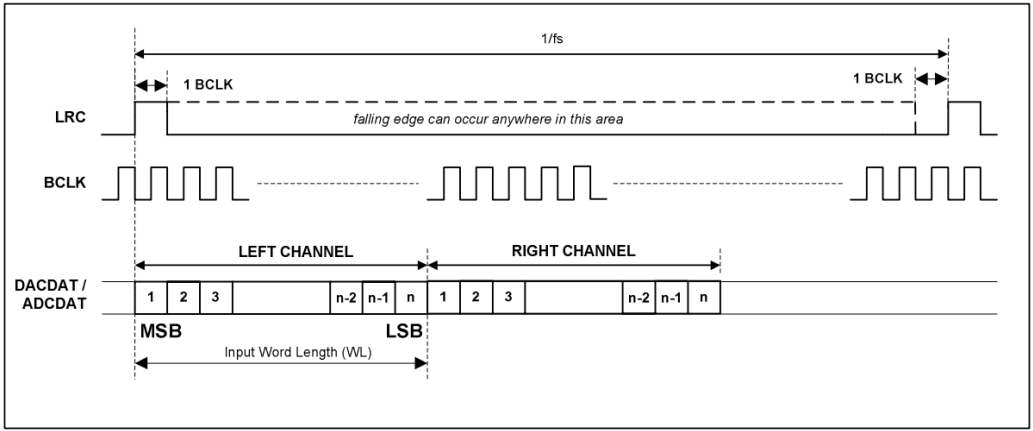
Digital formats - I2S



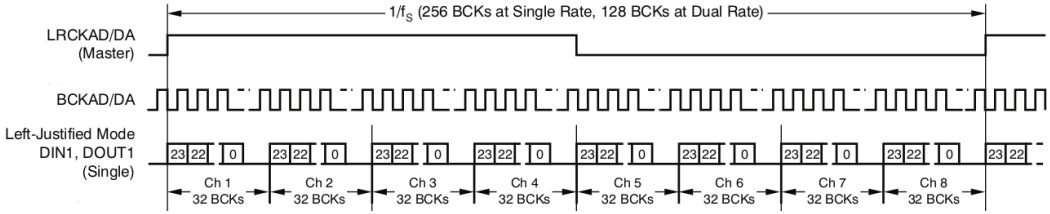
Digital formats - DSP A



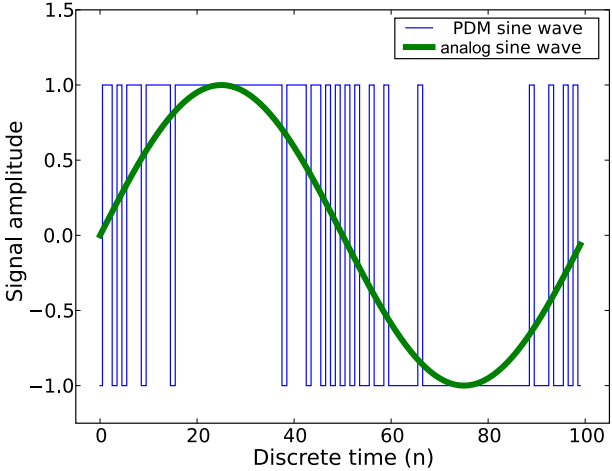
Digital formats - DSP B



Digital formats - TDM



Two signals per channels, clock and data



ASoC

ASoC, ALSA System on Chip: is a Linux kernel subsystem created to provide better ALSA support for system-on-chip and portable audio codecs. It allows to reuse codec drivers across multiple architectures and provides an API to integrate them with the SoC audio interface.

- created for that use case
- designed for codec drivers reuse
- has an API to write codec drivers
- has an API to write SoC interface drivers

ASoC components

- Codec class drivers: define the codec capabilities (audio interface, audio controls, analog inputs and outputs).
- Platform class drivers: defines the SoC audio interface (also referred as CPU DAI), sets up DMA when applicable.
- Codec to platform integration: nowadays, usually done through device tree, previously required writing a machine driver in C. See my 2016 ELC-E talk.

Note: The codec can be part of another IC (PMIC, Bluetooth or MODEM chips).

simple-card

Most sound cards, can now be described using device tree. This is done using a sound node with a `simple-audio-card` compatible string.

- The DT bindings are documented in `Documentation/devicetree/bindings/sound/simple-card.yaml`
- The driver handling it is `sound/soc/generic/simple-card.c`

Since 2017, OF-graph based bindings are available.

- They are documented in `Documentation/devicetree/bindings/sound/audio-graph-card.txt`
- The driver handling it is `sound/soc/generic/audio-graph-card.c`

Both required a few changes in the SoC DAI drivers to be usable for example to select the audio mode for the SSC on Microchip SoCs or configure properly the i.MX audmux.

simple-card - example 1

Let's say we have an ADAU1372 codec connected to an i.Mx6UL SAI. First, enable the SAI and the codec:

```
&sai2 {
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_sai2>;
        status = "okay";
};

&i2c1 {
        adau1372: codec@3c {
                #sound-dai-cells = <0>;
                compatible = "adi,adau1372";
                reg = <0x3c>;
                clock-names = "mclk";
                clocks = <&adau1372z_xtal>;
        };
};

/ {
        adau1372z_xtal: adau1372z_xtal {
                compatible = "fixed-clock";
                #clock-cells = <0>;
                clock-frequency = <12288000>;
        };
};
```

simple-card - example 1

Now, describe the sound card:

```
sound {
        compatible = "simple-audio-card";
        simple-audio-card,name = "imx6ul-adau1372";

        simple-audio-card,dai-link@0 {
                format = "i2s";
                bitclock-master = <&adau1372_dai>;
                frame-master = <&adau1372_dai>;

                sai2_dai: cpu {
                        sound-dai = <&sai2>;
                };

                adau1372_dai: codec {
                        sound-dai = <&adau1372>;
                };
        };
};
```

For convenience, the codec is the master, it generates both BCLK and FSCLK.

simple-card - example 2

The ADAU1372 has actually 4 channels and can do TDM:

```
sound {
        compatible = "simple-audio-card";
        simple-audio-card,name = "imx6ul-adau1372";

        simple-audio-card,dai-link@0 {
                format = "i2s";
                bitclock-master = <&adau1372_dai>;
                frame-master = <&adau1372_dai>;

                sai2_dai: cpu {
                        sound-dai = <&sai2>;
                        dai-tdm-slot-num = <4>;
                        dai-tdm-slot-width = <32>;
                };

                adau1372_dai: codec {
                        sound-dai = <&adau1372>;
                        dai-tdm-slot-num = <4>;
                        dai-tdm-slot-width = <32>;
                };
        };
};
```

However, the ADAU1372 has an hardware issue and doesn't generate the proper BCLK when doing TDM4 with a 32kHz sample rate. The SAI has to be master:

```
sound {
        compatible = "simple-audio-card";
        simple-audio-card,name = "imx6ul-adau1372";

        simple-audio-card,dai-link@0 {
                format = "i2s";
                bitclock-master = <&sai2_dai>;
                frame-master = <&sai2_dai>;

                sai2_dai: cpu {
                        sound-dai = <&sai2>;
                        dai-tdm-slot-num = <4>;
                        dai-tdm-slot-width = <32>;
                };

                adau1372_dai: codec {
                        sound-dai = <&adau1372>;
                        dai-tdm-slot-num = <4>;
                        dai-tdm-slot-width = <32>;
                };
        };
};
```

simple-card - example 3

The result is not what is expected:

```
# aplay test.wav
Playing WAVE 'test.wav' :
Signed 16 bit Little Endian, Rate  32000 Hz, Stereo
aplay: set_params:1403: Unable to install hw params:
[...]
# dmesg
[...]
fsl-sai 202c000.sai: failed to derive required Tx rate: 4096000
fsl-sai 202c000.sai: ASoC: can't set 202c000.sai hw params: -22
# cat /sys/kernel/debug/clk/clk_summary
pll3                        1        1        0   480000000          0     0  50000
   pll3_bypass              1        1        0   480000000          0     0  50000
      pll3_usb_otg          2        3        0   480000000          0     0  50000
         pll3_pfd2_508m     0        0        0   508235294          0     0  50000
            sai2_sel        0        0        0   508235294          0     0  50000
               sai2_pred    0        0        0   127058824          0     0  50000
                  sai2_podf 0        0        0    63529412          0     0  50000
                     sai2   0        0        0    63529412          0     0  50000
```

Indeed, there is now way for the SAI to divide 63529412 to get the proper BCLK!

It is possible to reparent clocks using `assigned-clock-parents` and set the clock rate using `assigned-clock-rates`.

```
&sai2 {
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_sai2>;
        assigned-clocks = <&clks IMX6UL_CLK_SAI2_SEL>, <&clks IMX6UL_CLK_SAI2>;
        assigned-clock-parents = <&clks IMX6UL_CLK_PLL4_AUDIO_DIV>;
        assigned-clock-rates = <196608000>, <24576000>;
        status = "okay";
};
```

Notice that 24.576MHz was selected for the sai input clock as it is not able to divide by 3 to obtain the 4.096MHz BCLK.

There is a possible cost reduction, the SAI is able to output its clock to feed to the codec MCLK instead of the crystal:

```
&sai2 {
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_sai2>;
        fsl,sai-mclk-direction-output;
        status = "okay";
};

&i2c1 {
        adau1372: codec@3c {
                #sound-dai-cells = <0>;
                compatible = "adi,adau1372";
                reg = <0x3c>;
                clock-names = "mclk";
                clocks = <&clks IMX6UL_CLK_SAI2>;
                assigned-clocks = <&clks IMX6UL_CLK_SAI2_SEL>, <&clks IMX6UL_CLK_SAI2>;
                assigned-clock-parents = <&clks IMX6UL_CLK_PLL4_AUDIO_DIV>;
                assigned-clock-rates = <196608000>, <24576000>;
        };
};
```

This replaces the 12.288MHz crystal by the 24.576 MCLK from the SAI. This works because the codec has a configurable divider for MCLK and can divide by 2. Also the clock parents and rates assignment has moved to the codec because of probing order.

simple-card - routing

It is possible but not mandatory to list the actual audio connections present on the board, this is called routing. The first step is to define the board connectors, in this case two stereo line input jack (Line0 and Line1) and a stereo jack output.

```
simple-audio-card,widgets =
        "Line", "Line0",
        "Line", "Line1",
        "Headphone", "Headphone Jack",
```

simple-card - routing

Routing audio from the codec to the board connector is then done using `simple-audio-card,routing`

```
simple-audio-card,routing =
        "AIN0", "Line0",
        "AIN1", "Line0",
        "AIN2", "Line1",
        "AIN3", "Line1",
        "Headphone Jack", "HPOUTL",
        "Headphone Jack", "HPOUTR",
```

Look for `SND_SOC_DAPM_OUTPUT` and `SND_SOC_DAPM_INPUT` to know what the codec is providing.

Amplifier

What about the amplifier?

- Supported using *auxiliary devices*
- There is a driver for simple amplifiers driven by a single GPIO, `simple-amplifier`.

simple-amplifier

```
dio2133: analog-amplifier {
        compatible = "simple-audio-amplifier";
        sound-name-prefix = "AU2";
        VCC-supply = <&hdmi_5v>;
        enable-gpios = <&gpio GPIOH_5 GPIO_ACTIVE_HIGH>;
};

sound {
        compatible = "amlogic,gx-sound-card";
        model = "GXL-LIBRETECH-S905X-CC";
        audio-aux-devs = <&dio2133>;
        audio-widgets = "Line", "Lineout";
        audio-routing = "AU2 INL", "ACODEC LOLN",
                        "AU2 INR", "ACODEC LORN",
                        "Lineout", "AU2 OUTL",
                        "Lineout", "AU2 OUTR";
```

Audio is routed through AU2, the amplifier.

Troubleshooting: no sound

Audio seems to play for the correct duration but there is no sound:

- Unmute `Master` and the relevant controls
- Turn up the volume
- Check the codec analog muxing and mixing (use alsamixer)
- Check the amplifier configuration
- Check the routing

Troubleshooting: no sound

When trying to play sound but it seems stuck:

- Check pinmuxing
- Check the configured clock directions
- Check the master/slave configuration
- Check the clocks using an oscilloscope
- Check pinmuxing
- Some SoCs also have more muxing (NXP i.Mx AUDMUX, TI McASP)

Troubleshooting: write error

```
# aplay test.wav
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
aplay: pcm_write:1737: write error: Input/output error
```

- Usually caused by an issue in the routing
- Check that the codec driver exposes a stream named "Playback"
- Use `vizdapm`: `https://github.com/mihais/asoc-tools`

Troubleshooting: over/underruns

```
# aplay test.wav
Playing WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
underrun!!! (at least 1.899 ms long)
underrun!!! (at least 0.818 ms long)
underrun!!! (at least 2.912 ms long)
underrun!!! (at least 8.558 ms long)
```

- Usually caused by an imprecise BCLK
- Try to find a better PLL and dividers combination

Troubleshooting: going further

- Have a look at the CPU DAI driver and its callback. In particular: `.set_clkdiv` and `.set_sysclk` to understand how the various clock dividers are setup. `.hw_params` or `.set_dai_fmt` may do some muxing
- Have a look at the codec driver callbacks, `.set_sysclk` as the `clk_id` parameter is codec specific.
- Remember using a codec as slave is an uncommon configuration and is probably untested.
- When in doubt, use `devmem` or `i2cget`

Going further

Once the kernel and device tree part are working, it is often necessary to split multiple channel inputs and outputs in individual channels or pair of channels. This can be done efficiently in userspace using alsalib, more on that on our blog: `https://bootlin.com/blog/audio-multi-channel-routing-and-mixing-using-alsalib/`

References

- Documentation/sound/alsa/soc/
- Common Inter-IC Digital Interfaces for Audio Data Transfer by Jerad Lewis, Analog Devices, Inc. http://www.analog.com/media/en/technical-documentation/technical-articles/MS-2275.pdf?doc=an-1327.pdf
- I²S specification https://web.archive.org/web/20060702004954/http://www.semiconductors.philips.com/acrobat_download/various/I2SBUS.pdf

Questions? Suggestions? Comments?

Alexandre Belloni
alexandre.belloni@bootlin.com

http://bootlin.com/pub/conferences/2020/elc/belloni-alsa-asoc/