Application Note

Sitara SoC 上的 Linux 音频

Suren Porwar

摘要

McASP 是 TI 的多通道音频串行端口。McASP 作为通用音频串行端口的功能针对多通道、多区域音频应用的需求进行了优化。

它被设计到许多音频产品中，如音频/视频接收器、条形音箱、汽车放大器和信息娱乐系统。

本应用报告面向不熟悉如何使用 Linux 作为 HLOS（高级操作系统）来设计 McASP 音频系统的工程师。

内容

商标

1 简介

McASP 最有用的一点之一是其灵活的时钟方案，该方案可实现接收端口和发送端口之间的完全独立，但这种灵活性意味着使用 McASP 实现音频系统的工程师必须做出一些重要的设计选择。

本文档的主要目标是使工程师能够更轻松地确定如何将 McASP（或多个 McASP）连接到其系统中的音频器件。

各种 TI SoC 上的音频子系统包含两个主要元件：

1. 多通道音频串行端口 (McASP) - 通过 IC 间音频 (I2S) 等业界通用协议在主机处理器与外部音频外设（例如编解码器）之间提供全双工串行接口。
2. 系统 DMA 引擎 - 为 McASP 提供对系统内存的直接访问，以从中读取音频样本（用于播放）或将音频样本存储到（用于捕获）。

除了上述功能外，大多数 TI EVM 和入门套件还具有连接到板载编解码器的线路输入/输出插孔，该编解码器可以在模拟信号和 McASP 支持的数字协议之间进行转换。

首先，几个免责声明：

- McASP 通常用于使用时分多路复用 (TDM) 协议与器件连接，在大多数情况下，这些器件将使用称为 IC 间音频 (I2S) 的特定 TDM 配置。假设在本文档提供的所有示例中都使用 I2S，但这有点武断。使用多时隙 TDM 或使用 I2S 将 McASP 物理连接到器件是相同的。
- 本文档并不旨在提供详细的规格信息。此外，本文档旨在涵盖一般 McASP，而不是特定 TI 器件上的 McASP。它简要介绍了 McASP 的主要特性，但器件特定的技术参考手册 (TRM) 仍然是获取架构和芯片级详细信息的理想场所。

2 软件架构

高级 Linux 声音架构 (ALSA) 现在是 Linux 系统中最流行的架构，可为 Linux 操作系统提供音频和 MIDI 功能。

ALSA 具有以下重要特性：

- 高效支持从消费类声卡到专业多通道音频接口的所有类型的音频接口。
- 完全模块化的声音驱动程序。
- SMP 和线程安全设计。
- 用户空间库 (alsa-lib) 可简化应用程序编程并提供更高级别的功能。
- 支持较旧的 Open Sound System (OSS) API，为大多数 OSS 程序提供二进制兼容性。

ALSA 片上系统 (ASoC) 层专为 SoC 音频而设计。ASoC 层的总体项目目标为嵌入式片上系统处理器和便携式音频编解码器提供更好的 ALSA 支持。

ASoC 层还提供以下特性：

- 编解码器独立性。允许在其他平台和计算机上重复使用编解码器驱动程序。
- 在编解码器和 SoC 之间轻松设置 I2S/PCM 音频接口。每个 SoC 接口和编解码器都在内核中注册其音频接口功能。
- 动态音频电源管理 (DAPM)。DAPM 是一种 ASoC 技术，旨在最大限度地降低音频子系统的功耗、无论采用何种音频用例。DAPM 始终保证最低的音频功率状态，并且对用户空间音频组件完全透明。DAPM 是移动器件或具有复杂音频要求的器件的理想选择。
- 减少爆裂声和咔嗒声。通过按正确的顺序（包括使用数字静音）为编解码器上电/断电，可以减少爆裂声和咔嗒声。ASoC 在更改电源状态时向编解码器发送信号。
- 机器专用控制装置。允许机器向声卡添加控件，例如扬声器放大器的音量控制。

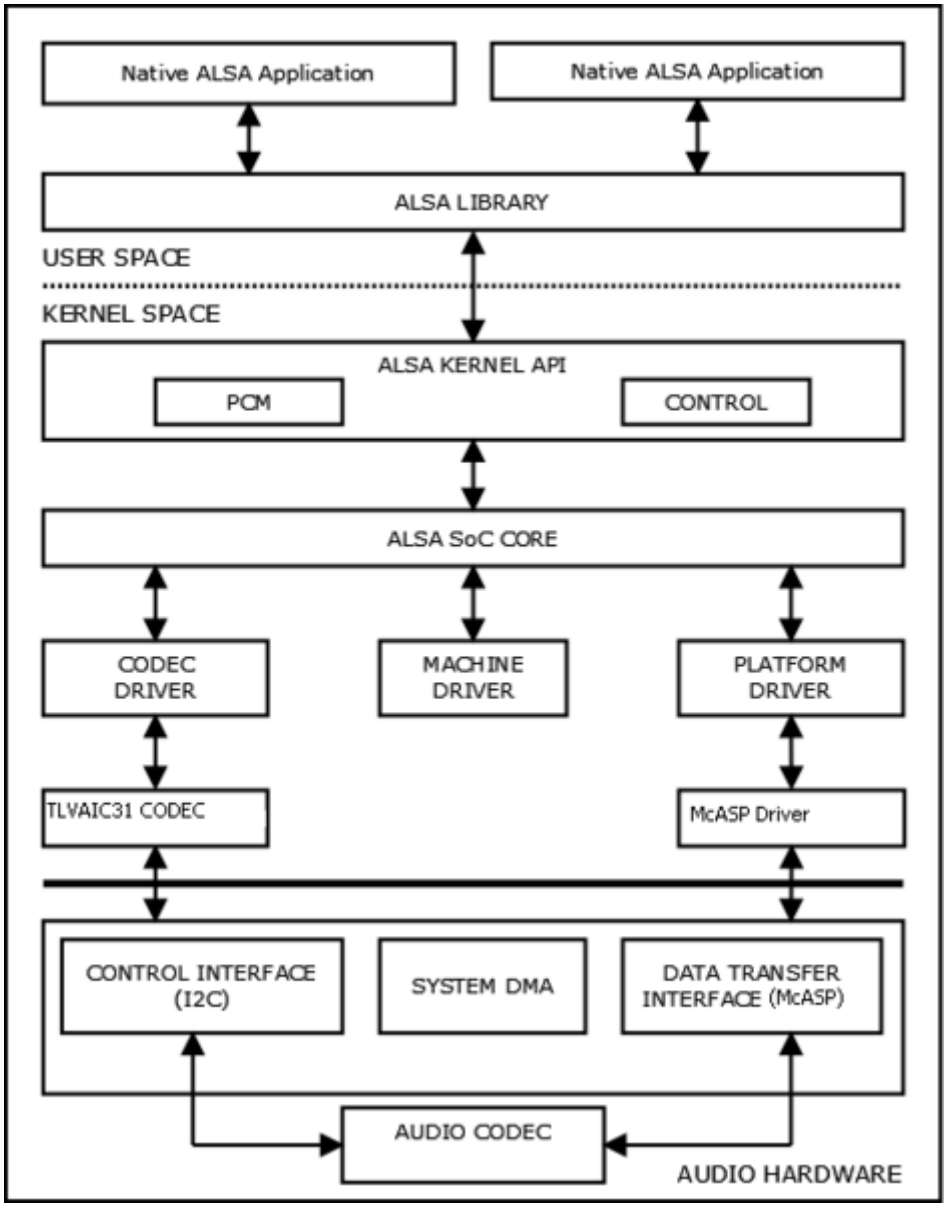

图 2-1. ASOC 架构

ASoC 将嵌入式音频系统拆分为多个可重用组件驱动程序：

- 编解码器类驱动程序：编解码器类驱动程序与平台无关，包含音频控制、音频接口功能、编解码器 DAPM 定义和编解码器 IO 功能。如果需要，该类别可扩展到 BT、FM 和调制解调器 IC。编解码器类驱动程序应该是可在任何架构和计算机上运行的通用代码。
- 平台类驱动程序：平台类驱动程序包括音频 DMA 引擎驱动程序、数字音频接口 (DAI) 驱动程序（例如 I2S、AC97、PCM）以及该平台的任何音频 DSP 驱动程序。
- 机器类驱动程序：机器类驱动程序就如同胶水，用于描述其他组件驱动程序并将其绑定在一起以形成 ALSA “声卡器件”。它处理任何机器特定控件和机器级别音频事件（例如在播放开始时打开放大器）。

有关 ASoC 的更多详细信息，请参见 Linux 操作系统源树中 Linux 内核文档，网址为 linux/Documentation/sound/alsa/soc，以及 www.alsa-project.org/main/index.php/ASoC。

3 声卡信息

注册的声卡信息可以使用命令 avplay -l 和 arecord -l 按如下方式列出。

例如，立体声声卡注册为卡 0。

```
root@am62axx-evm:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: AM62AxSKEVM [AM62Ax-SKEVM], device 0: 2b10000.audio-controller-tlv320aic3x-hifi tlv320aic3x-hifi-0 [2b10000.audio-controller-tlv320aic3x-hifi tlv320aic3x-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

ALSA 混音器控件用于设置录制路径上的回放音量和增益：

```
amixer -c 0 cset numid=71 on
amixer -c 0 cset numid=70 on
amixer -c 0 cset numid=64 on
amixer -c 0 cset numid=71 on
amixer -c 0 cset numid=70 on
amixer -c 0 cset numid=64 on
amixer -c 0 cset numid=65 on
amixer -c 0 cset numid=15 127
```

4 McASP - 外部信号

- 数据引脚：McASP 可能具有多达 16 个串行器。这些串行器连接到数据引脚，称为 AXR 引脚。它们被命名为这样（“音频发送/接收”=“AXR”）是因为任何 McASP 数据引脚都可以配置为用作输入或输出。请注意，如果 AXR 引脚作为发送器运行，则需要重新初始化 McASP，以将其重新配置为接收器。运行期间不支持动态切换方向。McASP 数据引脚连接非常简单。由于任何 McASP AXR 引脚都可以配置为发送或接收，因此设计人员可以自由选择系统最方便的引脚。
- 静音引脚：McASP 可以具有多达两个静音引脚：
- AMUTEIN – 这是输入。一些外部器件具有静音输出引脚；这样的信号可以连接到 AMUTEIN。在这种情况下，McASP 可配置为使 I2S 输出静音。
- AMUTE – 这是 McASP 可在特定错误条件下驱动的输出。有关更多详细信息，请参阅特定于器件的 TRM。静音引脚连接也很简单。AMUTE 引脚的行为配置需要一些规划（请参阅器件特定 TRM），但很容易确定其连接位置。如果下游器件有一个静音输入引脚，那么这就是 AMUTE 应连接的引脚。
- 时钟引脚：McASP 可以具有多达六个时钟引脚，所有（以及任何）引脚都可以在内部生成或从外部提供。McASP 上有三种类型的时钟信号：高频 (AHCLK)、位 (ACLK) 和帧同步 (AFS)。但是，每种类型都有一个传输版本（例如 X，AHCLKX）和一个接收版本（例如 R，AFSR）。每个 McASP 都有一个接收时钟部分和一个发送时钟部分。这些时钟有时称为接收端口和发送端口，每个时钟端口构成一个时钟区域。因此，每个 McASP 都有两个潜在的时钟区域。它们可以相互异步运行，但在某些情况下，适合同步运行。
- AHCLKX 和 AHCLKR – 这些是高频时钟引脚，有时称为主时钟（通常在音频编解码器上称为 MCLK）。McASP 使用主时钟只有一个目的：对其进行分频并生成一个位时钟。在一些情况下，不需要主时钟。
- ACLKX 和 ACLKR – 这些是位时钟，通常在音频器件上称为 BCK。数据是相对于位时钟边沿进行输入和输出的。此外，McASP 的许多内部逻辑（状态机等）都由位时钟运行，因此始终需要位时钟。
- AFSX 和 AFSR – 这些是帧同步时钟，通常称为字时钟，更常见的称呼是左右时钟 (LRCK)。“左右”术语来自 I2S 格式的立体声音频，其中帧同步时钟的边沿表示与左右通道相对应的位。帧同步时钟以音频流的采样率运行。

下图是 McASP 外部信号的简化表示，省略了静音引脚。

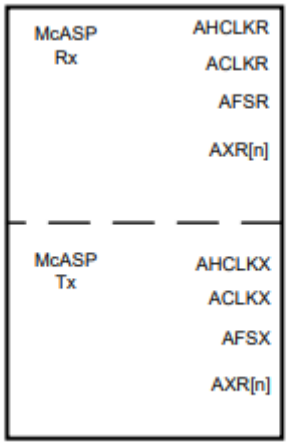

图 4-1. MCASP 外部信号

5 MCASP 时钟生成和配置

在某些情况下，MCASP 需要充当时钟主器件，并且必须生成时钟。

下图是 MCASP 时钟生成架构的简化方框图。有关更多详细信息，请参阅特定于器件的 TRM。

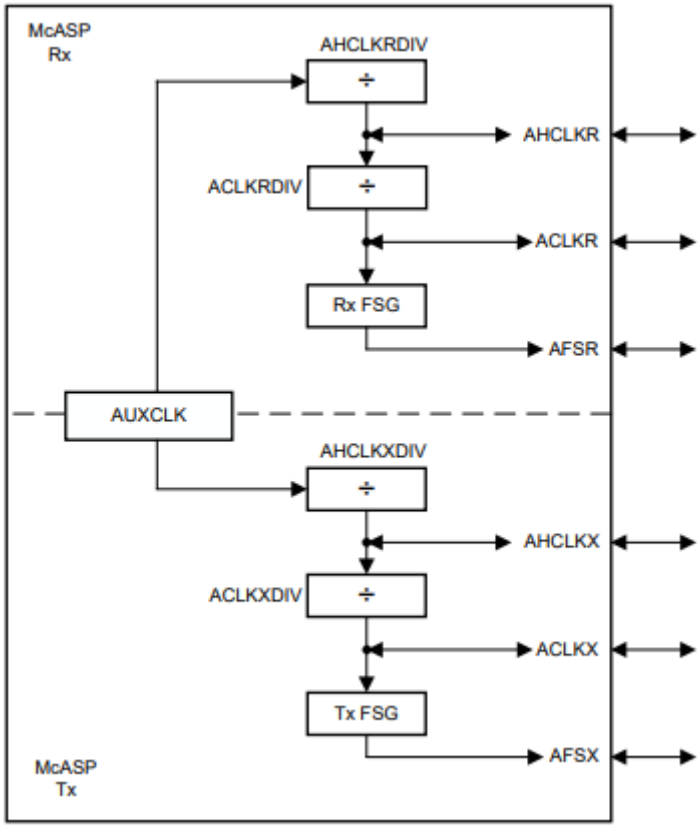

图 5-1. MCASP 时钟生成架构

图 2 中标记为 AUXCLK 的块是 McASP 的 Rx 和 Tx 部分均可用的时钟源。AUXCLK 的源因器件而异，但通常直接源自器件的主系统时钟源（通常是板载振荡器或外部生成的方波）。

对于接收和发送部分，AUXCLK 被馈送到整数高频时钟分频器中：AHCLKRDIV 或 AHCLKXDIV。该时钟信号可以进行分频以生成 McASP 的 Rx 或 Tx 主时钟：AHCLKR 或 AHCLKX。如果配置为该功能，则可在其相应引脚上驱动此信号。它还馈送下一个时钟分频器：ACLKRDIV 或 ACLKXDIV。同样，整数位时钟分频器 ACLK[R/X]DIV 生成 McASP 的位时钟：ACLKR 或 ACLKX。该信号可在相应引脚上驱动出来。请注意，位时钟分频器可改为由 AHCLK[R/X] 引脚馈送。时钟生成的下一个阶段看起来类似。这些是接收和发送帧同步发生器 (FSG)。FSG 的输出将是 AFSR 和 AFSX。这些也可以从其相应的时钟引脚中驱动。帧同步发生器可由提供给 ACLK[R/X] 引脚的外部位时钟馈送。这种情况并不常见，因为位时钟和帧同步通常都在内部生成，也可能同时在外部生成。

关于 McASP 的时钟生成架构的一个关键点是，McASP 具有可用于生成内部时钟的整数分频器（和帧同步发生器），并可从其相应的器件引脚驱动这些内部时钟。

6 虚拟声卡 DTS 更改

要在 DTS 中使用虚拟编解码器，您需要应用以下补丁：

https://patchwork.kernel.org/project/alsa-devel/patch/5652E348.8080002@invoxia.com/

器件树更改为包括虚拟编解码器和注册为虚拟声卡。

```
codec_test: codec_test {
compatible = "linux,snd-soc-dummy";
#sound-dai-cells = <0>;
status="okay";
};

codec_test: codec_test {
                compatible = "linux,snd-soc-dummy";
                #sound-dai-cells = <0>;
                status="okay";
        };



codec_audio: sound {

                compatible = "simple-audio-card";
                simple-audio-card,name = "AM62X-DUMMY";
                simple-audio-card,format = "i2s";
                simple-audio-card,bitclock-master = <&sound_master0>;
                simple-audio-card,frame-master = <&sound_master0>;



                sound_master0: simple-audio-card,cpu {
                sound-dai = <&mcasp1>;
                system-clock-direction-out;
                };



                simple-audio-card,codec {
                        sound-dai = <&codec_test>;
                };
        };
```

7 单 DAI 链路或单声卡

```
-----------------------------------------
Example 1 - single DAI link:
-----------------------------------------

sound {
    compatible = "simple-audio-card";
    simple-audio-card,name = "VF610-Tower-Sound-Card";
    simple-audio-card,format = "left_j";
    simple-audio-card,bitclock-master = <&dailink0_master>;
    simple-audio-card,frame-master = <&dailink0_master>;
    simple-audio-card,widgets =
        "Microphone", "Microphone Jack",
        "Headphone", "Headphone Jack",
        "Speaker", "External Speaker";
    simple-audio-card,routing =
        "MIC_IN", "Microphone Jack",
        "Headphone Jack", "HP_OUT",
        "External Speaker", "LINE_OUT";

    simple-audio-card,cpu {
        sound-dai = <&sh_fsi2 0>;
    };

    dailink0_master: simple-audio-card,codec {
        sound-dai = <&ak4648>;
        clocks = <&osc>;
    };
};
```

8 多 DAI 链路 - 单卡但多个子器件

```
-----------------------------------------
Example 2 - many DAI links:
-----------------------------------------

sound {
    compatible = "simple-audio-card";
    simple-audio-card,name = "Cubox Audio";

    simple-audio-card,dai-link@0 {      /* I2S - HDMI */
        reg = <0>;
        format = "i2s";
        cpu {
            sound-dai = <&audio1 0>;
        };
        codec {
            sound-dai = <&tda998x 0>;
        };
    };

    simple-audio-card,dai-link@1 {      /* S/PDIF - HDMI */
        reg = <1>;
        cpu {
            sound-dai = <&audio1 1>;
        };
        codec {
            sound-dai = <&tda998x 1>;
        };
    };

    simple-audio-card,dai-link@2 {      /* S/PDIF - S/PDIF */
        reg = <2>;
        cpu {
            sound-dai = <&audio1 1>;
        };
        codec {
            sound-dai = <&spdif_codec>;
        };
    };
};
```

参考资料：https://www.kernel.org/doc/Documentation/devicetree/bindings/sound/simple-card.txt

9 MCASP - 实际示例

McASP 通常连接到具有第 3 节中所述相同类型的时钟引脚（主时钟，位和帧同步）的音频器件，因此很容易确定连接到另一个音频器件的时钟引脚的 McASP 时钟引脚类型：位时钟转到位时钟、帧同步转到帧同步等。有关 McASP 的大多数问题都与使用哪个时钟端口（接收或发送）以及应为哪个方向（输入或输出）配置引脚有关。使用 McASP 解决音频设计这方面问题的最简单方法是使用一些实际示例。

对于不熟悉数字音频系统设计的工程师，可能必须再次考虑是将其音频器件挂钩到 McASP 的接收端口还是发送端口。首先要考虑的是数据的传输方向。如果 McASP 将从音频器件接收数据，则该器件的时钟引脚应连接到 McASP 的接收时钟引脚。

10 McASP 作为接收器

条形音箱或汽车立体声等音频系统通常具有模拟输入。在所有这些情况下，都需要音频 ADC。一旦发生模数转换，必须将生成的数字数据流馈送到 McASP。由于 McASP 从 ADC 接收数据，因此 McASP 的 Rx 时钟引脚将连接到 ADC 的时钟引脚。

10.1 ADC 或编解码器作为时钟主器件

如果 ADC 必须配置为时钟主器件，则它将是生成所有时钟的器件。已确定器件将连接到 McASP 的 Rx 时钟引脚，如下图所示。

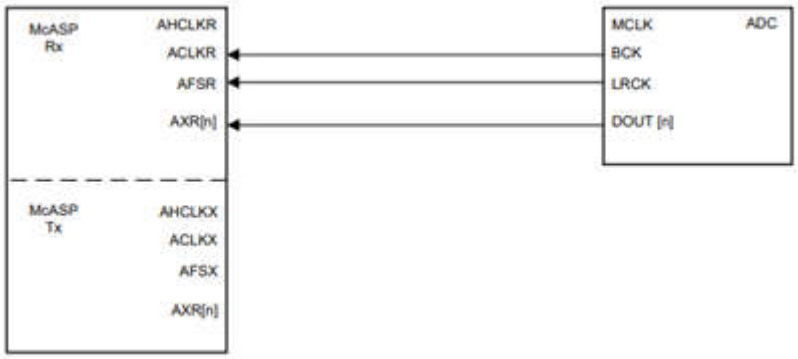

图 10-1. ADC 作为时钟主器件

10.2 器件树更改 - 编解码器作为主器件，MCASP 作为从器件

```
tlv320_mclk: clk-0 {
    #clock-cells = <0>;
    compatible = "fixed-clock";
    clock-frequency = <12288000>;
};

codec_audio: sound {
    compatible = "simple-audio-card";
    simple-audio-card,name = "AM62x-SKEVM";
    simple-audio-card,widgets =
        "Headphone",    "Headphone Jack",
        "Line",     "Line In",
        "Microphone",   "Microphone Jack";
    simple-audio-card,routing =
        "Headphone Jack",   "HPLOUT",
        "Headphone Jack",   "HPROUT",
        "LINE1L",       "Line In",
        "LINE1R",       "Line In",
        "MIC3R",        "Microphone Jack",
        "Microphone Jack",  "Mic Bias";
    simple-audio-card,format = "dsp_b";
    simple-audio-card,bitclock-master = <&sound_master>;
    simple-audio-card,frame-master = <&sound_master>;
    simple-audio-card,bitclock-inversion;

    simple-audio-card,cpu {
        sound-dai = <&mcasp1>;
    };

    sound_master: simple-audio-card,codec {
        sound-dai = <&tlv320aic3106>;
        clocks = <&tlv320_mclk>;
    };
};

tlv320aic3106: audio-codec@1b {
    #sound-dai-cells = <0>;
    compatible = "ti,tlv320aic3106";
    reg = <0x1b>;
```

```
    ai3x-micbias-vg = <1>;  /* 2.0V */
};
```

参考资料：https://git.ti.com/cgit/ti-linux-kernel/ti-linux-kernel/tree/arch/arm64/boot/dts/ti/k3-am62p5-sk.dts?h=ti-linux-6.12.y

需要注意的重要事项：

- McASP 的接收时钟引脚 ACLKR 和 AFSR 用作输入。ADC 作为时钟主器件驱动这些时钟线。McASP 必须配置为时钟从器件，这意味着 ACLKR 和 AFSR 引脚必须配置为输入，并且必须配置为外部时钟源。这两件事听起来是等效的，但实际上并不是。McASP 引脚方向和时钟源是两种不同的设置。有关如何配置的更多信息，请参阅器件特定 TRM。
- 不使用 AHCLKR，只需要位时钟和帧同步。仅当 McASP 的 Rx 部分被馈送到高频时钟时，才需要 AHCLKR，然后必须对其进行分频以生成位时钟和帧同步。在上面的示例中，这不是必需的，因为 McASP 不需要生成 Rx 时钟。

```
main_mcasp1_pins_default: main-mcasp1-default-pins {
        pinctrl-single,pins = <
            AM62PX_IOPAD(0x0090, PIN_INPUT, 2) /* (U24) GPMC0_BE0n_CLE.MCASP1_ACLKX */
            AM62PX_IOPAD(0x0098, PIN_INPUT, 2) /* (AA24) GPMC0_WAIT0.MCASP1_AFSX */
            AM62PX_IOPAD(0x008c, PIN_OUTPUT, 2) /* (T25) GPMC0_WEn.MCASP1_AXR0 */
            AM62PX_IOPAD(0x0084, PIN_INPUT, 2) /* (R25) GPMC0_ADVn_ALE.MCASP1_AXR2 */
        >;
    };

&mcasp1 {
    status = "okay";
    #sound-dai-cells = <0>;

    pinctrl-names = "default";
    pinctrl-0 = <&main_mcasp1_pins_default>;

    op-mode = <0>;          /* MCASP_IIS_MODE */
    tdm-slots = <2>;

    serial-dir = <  /* 0: INACTIVE, 1: TX, 2: RX */
           1 0 2 0
           0 0 0 0
           0 0 0 0
           0 0 0 0
    >;
};
```

11 MCASP 作为发送器

11.1 器件树更改 - 编解码器作为从器件，MCASP 作为主器件

许多 ADC 无法自行生成时钟，必须作为时钟从器件运行。由于 McASP 从 ADC 接收数据，因此宜将其 Rx 时钟引脚配置为输出，并生成将驱动到 ADC 时钟引脚的时钟。请记住，McASP 具有整数时钟分频器。如果器件的 AUXCLK 运行频率可用于在所需频率下生成所有必要的 McASP Rx 时钟，则 AHCLKR、ACLKR 和 AFSR 都将在内部生成，并全部配置为输出，如下图所示。

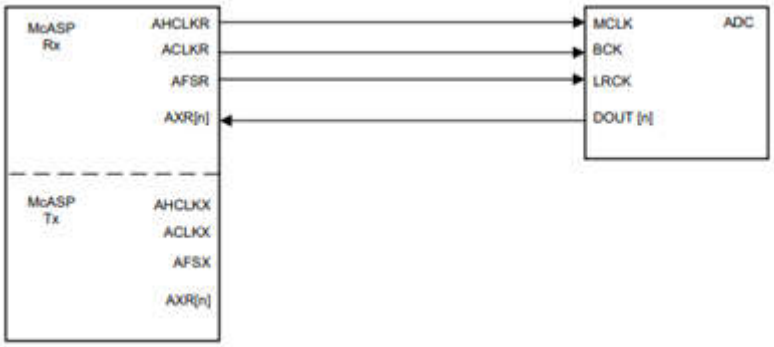

图 11-1. ADC 作为时钟从器件，所有 McASP 时钟均在内部生成

需要注意的重要事项：

- 所有 McASP Rx 时钟都是输出。
- AHCLKR 由 McASP 驱动。这是因为 ADC 通常需要高频时钟才能运行。如需了解更多信息，请参阅器件特定 ADC 数据手册。

```
clocks {
                /* 25MHz reference crystal */
                ref25: oscillator {
                        compatible = "fixed-clock";
                        #clock-cells = <0>;
                        clock-frequency = <24576000>;
            clock-output-names = "mclk_24m576";
                };
        };

        dep_audio: sound-tac5112 {
                compatible = "simple-audio-card";
                simple-audio-card,name = "AM62x-TAC5112-DEP";
                simple-audio-card,format = "i2s";
                simple-audio-card,bitclock-master = <&master>;
                simple-audio-card,frame-master  = <&master>;
        simple-audio-card,bitclock-inversion;

        simple-audio-card,widgets =
            "Line", "Line In",
            "Line", "Line Out";
        simple-audio-card,routing =
            "AIN1", "Line In",
            "Line Out", "OUT1";

                master: simple-audio-card,cpu {
                        sound-dai = <&mcasp0>;
            system-clock-direction-out;
                };

            master1: simple-audio-card,codec {
                        sound-dai = <&dep_codec>;
                };
        };
};

    main_mcasp0_pins_default: main-mcasp0-pins-default {
        pinctrl-single,pins = <
            /* Currently unused on the daughtercard */
```

```
            AM62X_IOPAD(0x01A8, PIN_OUTPUT, 0)  /* (D20) MCASP0_AFSR */
            AM62X_IOPAD(0x01A4, PIN_OUTPUT, 0)  /* (B20) MCASP0_ACLKR */
            AM62X_IOPAD(0x01A0, PIN_OUTPUT, 0)  /* (E18) MCASP0_AXR0 */
            AM62X_IOPAD(0x019C, PIN_INPUT, 0)  /* (B18) MCASP0_AXR1 */
        >;
    };

&mcasp0 {
    status = "okay";
    #sound-dai-cells = <0>;

    pinctrl-names = "default";
    pinctrl-0 = <&main_mcasp0_pins_default>;

    op-mode = <0>;          /* MCASP_IIS_MODE */
    tdm-slots = <2>;

    auxclk-fs-ratio = <2177>;

    serial-dir = <  /* 0: INACTIVE, 1: TX, 2: RX */
           1 2 0 0
           0 0 0 0
           0 0 0 0
           0 0 0 0
    >;
};
```

在上述情况下，时钟生成如下图所示：

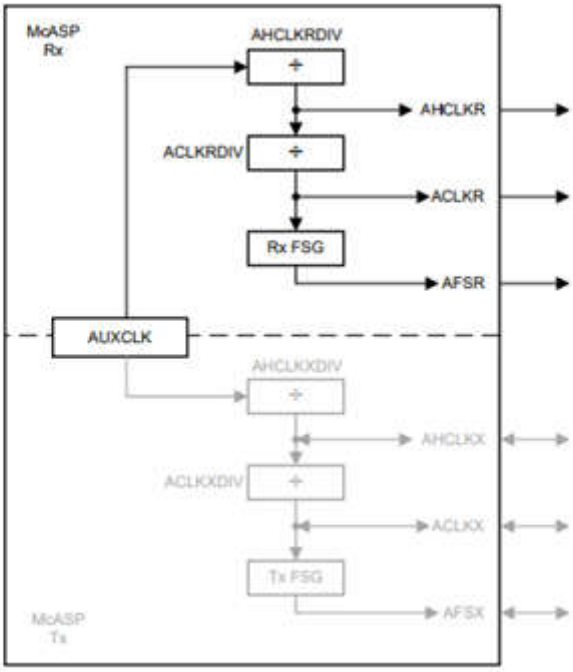

图 11-2. ADC 作为时钟从器件，Rx 时钟在内部生成

AUXCLK 馈送 AHCLKRDIV，而 AHCLKRDIV 馈送 ACLKRDIV，ACLKRDIV 馈送 Rx FSG。

参考：k3-am62-main.dtsi

```
mcasp0: audio-controller@2b00000 {
        compatible = "ti,am33xx-mcasp-audio";
        reg = <0x00 0x02b00000 0x00 0x2000>,
              <0x00 0x02b08000 0x00 0x400>;
        reg-names = "mpu", "dat";
        interrupts = <GIC_SPI 236 IRQ_TYPE_LEVEL_HIGH>,
                     <GIC_SPI 235 IRQ_TYPE_LEVEL_HIGH>;
        interrupt-names = "tx", "rx";
```

```
        dmas = <&main_bcdma 0 0xc500 0>, <&main_bcdma 0 0x4500 0>;
        dma-names = "tx", "rx";

        clocks = <&k3_clks 190 0>;
        clock-names = "fck";
        assigned-clocks = <&k3_clks 190 0>;
        assigned-clock-parents = <&k3_clks 190 2>;
        power-domains = <&k3_pds 190 TI_SCI_PD_EXCLUSIVE>;
        status = "disabled";
};

mcasp1: audio-controller@2b10000 {
        compatible = "ti,am33xx-mcasp-audio";
        reg = <0x00 0x02b10000 0x00 0x2000>,
              <0x00 0x02b18000 0x00 0x400>;
        reg-names = "mpu", "dat";
        interrupts = <GIC_SPI 238 IRQ_TYPE_LEVEL_HIGH>,
                     <GIC_SPI 237 IRQ_TYPE_LEVEL_HIGH>;
        interrupt-names = "tx", "rx";

        dmas = <&main_bcdma 0 0xc501 0>, <&main_bcdma 0 0x4501 0>;
        dma-names = "tx", "rx";

        clocks = <&k3_clks 191 0>;
        clock-names = "fck";
        assigned-clocks = <&k3_clks 191 0>;
        assigned-clock-parents = <&k3_clks 191 2>;
        power-domains = <&k3_pds 191 TI_SCI_PD_EXCLUSIVE>;
        status = "disabled";
};

mcasp2: audio-controller@2b20000 {
        compatible = "ti,am33xx-mcasp-audio";
        reg = <0x00 0x02b20000 0x00 0x2000>,
              <0x00 0x02b28000 0x00 0x400>;
        reg-names = "mpu", "dat";
        interrupts = <GIC_SPI 240 IRQ_TYPE_LEVEL_HIGH>,
                     <GIC_SPI 239 IRQ_TYPE_LEVEL_HIGH>;
        interrupt-names = "tx", "rx";

        dmas = <&main_bcdma 0 0xc502 0>, <&main_bcdma 0 0x4502 0>;
        dma-names = "tx", "rx";

        clocks = <&k3_clks 192 0>;
        clock-names = "fck";
        assigned-clocks = <&k3_clks 192 0>;
        assigned-clock-parents = <&k3_clks 192 2>;
        power-domains = <&k3_pds 192 TI_SCI_PD_EXCLUSIVE>;
        status = "disabled";
};
```

在许多情况下，AUXCLK 的运行频率不能分频为所需的位时钟和帧同步速率（请记住，McASP 只有整数分频器，有时计算结果无法满足需求）。在这种情况下，必须将某种适当的外部时钟源馈送到 AHCLKR 引脚，然后进行分频以生成位时钟和帧同步。图 7 展示了这一场景。

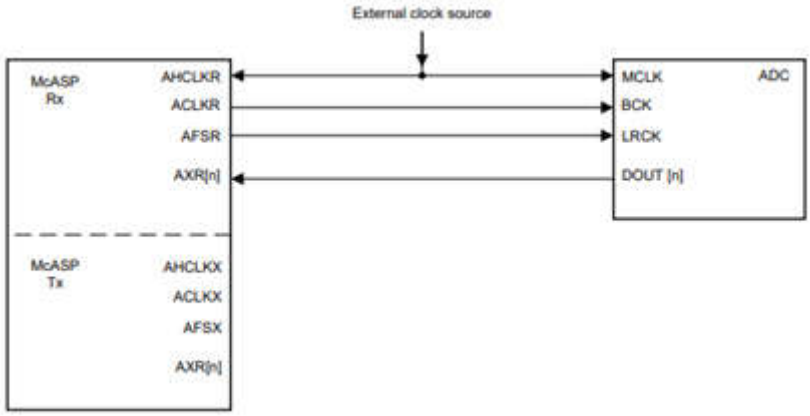

图 11-3. ADC 作为时钟从器件，外部主时钟

外部时钟同时馈送 ADC 的 MCLK 和 AHCLKR 引脚。下图展示了此场景的时钟生成。

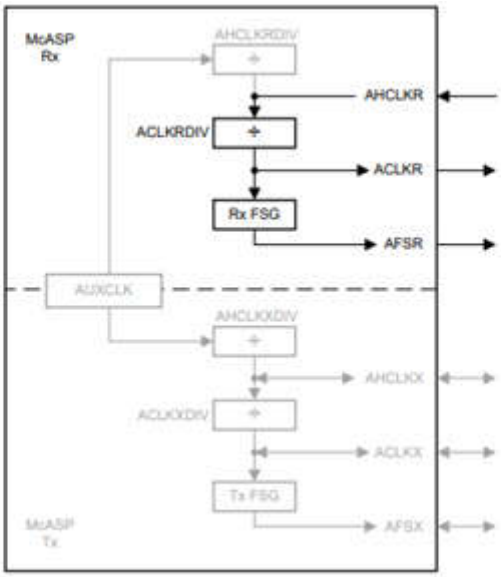

图 11-4. ADC 作为时钟从器件，外部主时钟用于 Rx 时钟生成

在这种情况下，不使用 AUXCLK；因此不使用 AHCLKRDIV。外部主时钟源用于生成 ACLKR 和 AFSR 信号，然后将这些信号馈送到 ADC 的 BCK 和 LRCK 引脚，以满足 ADC 充当时钟从器件的要求。

假设该外部时钟源向 AUDIO_EXT_REFCLK0 提供时钟并在 AM62x 上使用 MCASP2，则器件树需要添加以下线路：

```
    fixed_audio_refclk: clk-0 {
        status = "okay";

        #clock-cells = <0>;
        compatible = "fixed-clock";
        clock-frequency = <24576000>;
    };

    pinctrl-0 = <&board_pins_refclk>;

    board_pins_mcasp2: board-pins-mcasp2 {
        pinctrl-single,pins = <
            AM62X_IOPAD(0xf41d0, PIN_OUTPUT, 8) /* [Speaker_Audio_FSYNC] (A15)
UART0_CTSn.MCASP2_AFSX */
            AM62X_IOPAD(0xf41d4, PIN_OUTPUT, 8) /* [Speaker_Audio_CLK] (B15)
UART0_RTSn.MCASP2_ACLKX */
            AM62X_IOPAD(0xf41d8, PIN_OUTPUT, 8) /* [Speaker_Audio_OUT] (C15) MCAN0_TX.MCASP2_AXR0 */
        >;
    };

    board_pins_refclk: board-pins-refclk {
        pinctrl-single,pins = <
            AM62X_IOPAD(0xf4190, PIN_INPUT, 2) /* [Audio_EXT_Refclk0] (AE22)
RGMII2_RD3.AUDIO_EXT_REFCLK0 */
            AM62X_IOPAD(0xf41f0, PIN_INPUT, 0) /* [Ext_Refclk1] (A18) EXT_REFCLK1.EXT_REFCLK1 */
            AM62X_IOPAD(0xf413c, PIN_OUTPUT, 7) /* [Audio_Refclk_Enable] (AE18) RGMII1_TD2.GPIO0_77
*/
        >;
    };

&mcasp2 {
    status = "okay";
    #sound-dai-cells = <0>;

    /* CLOCKS:
     * BOARD_AUDIO_EXT_REFCLK0 <192 12> --> AHCLKR IN <192 9>
     * BOARD_AUDIO_EXT_REFCLK0 <192 18> --> AHCLKX IN <192 15>
     * REFCLK runs at 24,576 MHz
     */
```

```
    assigned-clocks        = <&k3_clks 192 9>, <&k3_clks 192 15>;
    assigned-clock-parents = <&k3_clks 192 12>, <&k3_clks 192 18>;
    assigned-clock-rates   = <24576000>, <24576000>;

    pinctrl-names = "default";
    pinctrl-0 = <&board_pins_mcasp2>;

    op-mode = <0>;          /* MCASP_IIS_MODE */
    tdm-slots = <2>;

    serial-dir = <  /* 0: INACTIVE, 1: TX, 2: RX */
           1 0 0 0
           0 0 0 0
           0 0 0 0
           0 0 0 0
    >;
    tx-num-evt = <0>;
    rx-num-evt = <0>;
};
```

12 参考资料

ALSA 链接

1. ALSA SoC 项目主页
2. ALSA 项目主页
3. ALSA 用户空间库
4. 使用 ALSA 音频 API 作者：Paul Davis

音频硬件编解码器

1. TLV320AIC31 - 具有耳机放大器的低功耗立体声编解码器
2. TLV320AIC3104 - 具有耳机放大器的低功耗立体声编解码器
3. TLV320AIC3111 - 具有嵌入式 miniDSP 和立体声 D 级扬声器放大器的低功耗立体声编解码器
4. TLV320AIC3106 - 低功耗立体声音频编解码器

Linux 内核绑定

1. https://www.kernel.org/doc/Documentation/devicetree/bindings/sound/simple-card.txt
2. https://www.kernel.org/doc/Documentation/devicetree/bindings/sound/davinci-mcasp-audio.txt
3. https://www.kernel.org/doc/Documentation/devicetree/bindings/clock/fixed-clock.txt
4. https://github.com/devicetree-org/dt-schema/blob/main/dtschema/schemas/clock/clock.yaml

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月