XR806 Audio PWM 驱动

使用指南

版本号：1.0

发布时间：2020-12-23

版本历史

版本	日期	责任人	版本描述
1.0	2020-12-23	AWA 1451	创建文档。

目录

表格目录

1 前言

1.1 文档简介

本文档介绍了 XR806 Audio PWM 驱动的使用方法。

1.2 目标读者

本文档适合于 XR806 音频驱动开发及维护的人员。

1.3 适用范围

此文档适用于 XR806 SDK，支持 XR806 系列芯片产品。

1.4 文档约定

1.4.1 标志说明

本文档采用各种醒目的标志来表示在操作过程中应该特别注意的地方，这些标志的含义如下：

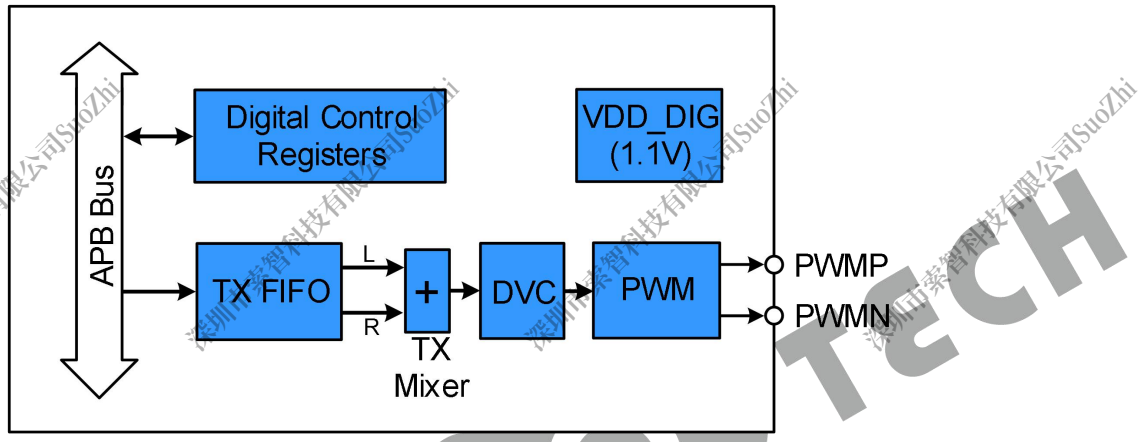
标识	说明
 **警告**	该标志后的说明应给予格外关注，如果不遵守，可能会导致人员受伤或死亡。
 **注意**	提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。
 **说明**	为准确理解文中指令、正确实施操作而提供的补充或强调信息。
 **窍门**	一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

2 概述

2.1 背景说明

XR806 Internal Audio PWM 是一款用于驱动 Class D 音频功放的数字差分 PWM 驱动器，通过将 PCM（Pulse Code Modulation，脉冲编码调制）音频转换成数字 PWM 调制输出 PWMP、PWMN 的差分信号，进而驱动 Class D 功放。

图 2-1 XR806 Internal Audio PWM 框图



2.2 规格特性

XR806 Audio PWM 驱动规格特性如下表所示。

表 2-1 XR806 Audio PWM 驱动规格特性表

驱动规格	规格描述	备注
声道数	支持播放声道数为 1、2 的 PCM 音频	
采样率	支持播放常见采样率 8KHz~48KHz 的 PCM 音频	
音量调节	支持播放音量-45~45dB 调节，3dB/Step	

2.3 文件位置

以 SDK 包为根目录，本驱动涉及到的主要文件位置如下。

表 2-2 Audio PWM 驱动的文件位置

驱动名	文件分类	文件位置
Audio PWM	源码文件	./src/driver/chip/codec/xradio_internal_codec.c
	头文件	./src/driver/chip/codec/xradio_internal_codec.h

3 应用说明

3.1 应用简述

Audio PWM 的代码已经内嵌到 XR806 SDK 中，通过配置以及函数接口调用即可使用。

3.2 配置说明

表 3-1 XR806 Audio PWM 驱动模块配置列表

配置项	配置说明
Audio PWM 功能使能	设置说明： 此项配置用于在 SDK 中启用 Audio PWN 功能，即在 platform init 阶段进行 Audio PWM 功能的注册。 设置位置： 工程目录下的 prj_config.h 文件，例如 wlan_demo 工程中的 /project/demo/audio_demo/prj_config.h 文件。 设置方式： 在 prj_config.h 文件，添加或修改以下定义，其中 1 为进行 Audio PWM 功能的注册，0 为不进行 Audio PWM 功能的注册。 /* Xradio internal codec sound card enable/disable */ #define PRJCONF_INTERNAL_SOUNDCARD_EN 1

3.3 接口说明

表 3-2 Audio PWM 驱动模板接口简介

接口名	简要介绍
基本读写寄存器接口	本接口用于 Audio PWM 硬件模块的寄存器读写操作，属于内部接口。 本接口在 xradio_internal_codec.c 文件中实现，并仅在此源文件中使用。
基本硬件配置接口	本接口用于 Audio PWM 硬件模块硬件参数的初始化，属于内部接口。
DMA 配置接口	本接口用于 Audio PWM 有关 DMA 的配置，属于内部接口。
Ioctl 接口	本接口用于 Audio PWM ioctl 的管理，属于内部接口。
codec_dai_ops 接口	本接口用于 Audio PWM 数字音频接口 DAI 的配置操作，属于内部接口。
codec_ops 接口	本接口用于 Audio PWM 打开、关闭等操作，属于内部接口。
codec_driver 接口	本接口用于 Audio PWM 驱动初始化，属于内部接口。
注册接口	本接口用于 Audio PWM 接口注册，属于外部接口。

接口名	简要介绍
	本接口在 xradio_internal_codec.c 文件中实现，并在 audio_arch.h 文件中声明。

3.3.1 Xradio_Codec_Priv 结构体

struct Xradio_Codec_Priv 结构体为 Audio PWM 驱动中使用的私有数据结构，其对应定义的全局指针变量为 xradio_codec_priv，在注册时会为其申请内存，具体定义如下代码段所示：

```
struct Xradio_Codec_Priv {
    //codec status contrl
    bool isCodecInit;
    bool isTxInit;
    volatile bool txRunning;

    //buffer control
    uint8_t *txBuf;
    uint8_t *writePointer;
    uint32_t txBufSize;

    //DMA control
    uint8_t *txDmaPointer;
    DMA_Channel txDMAChan;
    DMA_DataWidth tx_data_width;
    volatile uint8_t txHalfCallCount;
    volatile uint8_t txEndCallCount;

    //Semaphore control
    HAL_Semaphore txReady;
    bool isTxSemaphore;

    //misc control
    uint16_t tx_underrun_threshold;
    uint16_t cld_vol;
    uint8_t cld_ptn_sel;
};

static struct Xradio_Codec_Priv *xradio_codec_priv;
```

其结构体成员说明如下表所示

表 3-3 Xradio_Codec_Priv 结构体成员说明

信息项	说明
Codec 状态控制变	**isCodecInit**：标志 Codec 是否已经初始化，在 xradio_internal_codec_init 和

信息项	说明
量	xradio_internal_codec_deinit 中使用。 **isTxInit**：标志 Codec 是否已经播放 open；在 xradio_codec_open 和 xradio_codec_close 中配置。 **txRunning**：标志 Codec 是否已经 trigger DMA 播放，在 xradio_codec_dma_trigger 中配置。
buffer 控制变量	***txBuf**：播放 buffer 指针，在 xradio_codec_open 时申请 ***writePointer**：下一次播放写入 buffer 的指针 **txBufSize**：以 byte 为单位的播放 buffer 大小
DMA 控制变量	***txDmaPointer**：当前 DMA 播放搬运数据指针 **txDMAChan**：DMA 播放通道 **tx_data_width**：DMA 播放数据宽度 **TxHalfCallCount**：DMA 播放半中断计数 **txEndCallCount**：DMA 播放全中断计数
信号量控制变量	**txReady**：DMA 播放信号量 **isTxSemaphore**：DMA 播放信号量是否正在被占用标志
杂项控制	**tx_underrun_threshold**：播放 underrun 阈值配置变量 **cld_vol**：播放音量 **cld_ptn_sel**：播放 pattern 选择

3.3.2 常量数组

3.3.2.1 xradio_cld_clk_div[]

Audio PWM 不同采样率下 CLD 模块系统时钟分频比寄存器配置值映射关系数组，数组元素为 struct cld_clk_div 结构体常量；如下代码段所示：

```
struct cld_clk_div {
    uint32_t sample_rate; /* 采样率 */
    uint8_t  clk_div;     /* 分频比 */
};

static const struct cld_clk_div xradio_cld_clk_div[] = {
    {48000, 8},
    {24000, 16},
    {12000, 32},

    {32000, 12},
    {16000, 24},
    {8000,  48},

    {44100, 10},
```

```
    {22050, 20},
    {11025, 40},
};
```

3.3.2.2 xradio_cld_vol_gain[]

Audio PWM 不同播放增益与寄存器配置值映射关系数组，数组元素为 struct real_val_to_reg_val 结构体常量；如下代码段所示：

```
struct real_val_to_reg_val {
    uint32_t real_val; /*实际值 */
    uint32_t reg_val; /*寄存器值 */
};

static const struct real_val_to_reg_val xradio_cld_vol_gain[] = {
    {VOLUME_GAIN_MINUS_45dB, 4},
    {VOLUME_GAIN_MINUS_42dB, 8},
    {VOLUME_GAIN_MINUS_39dB, 12},
    {VOLUME_GAIN_MINUS_36dB, 16},
    {VOLUME_GAIN_MINUS_33dB, 20},
    {VOLUME_GAIN_MINUS_30dB, 24},
    {VOLUME_GAIN_MINUS_27dB, 28},
    {VOLUME_GAIN_MINUS_24dB, 32},
    {VOLUME_GAIN_MINUS_21dB, 36},
    {VOLUME_GAIN_MINUS_18dB, 40},
    {VOLUME_GAIN_MINUS_15dB, 44},
    {VOLUME_GAIN_MINUS_12dB, 48},
    {VOLUME_GAIN_MINUS_9dB,  52},
    {VOLUME_GAIN_MINUS_6dB,  56},
    {VOLUME_GAIN_MINUS_3dB,  60},

    {VOLUME_GAIN_0dB,  64},

    {VOLUME_GAIN_3dB,  68},
    {VOLUME_GAIN_6dB,  72},
    {VOLUME_GAIN_9dB,  76},
    {VOLUME_GAIN_12dB, 80},
    {VOLUME_GAIN_15dB, 84},
    {VOLUME_GAIN_18dB, 88},
    {VOLUME_GAIN_21dB, 92},
    {VOLUME_GAIN_24dB, 96},
    {VOLUME_GAIN_27dB, 100},
    {VOLUME_GAIN_30dB, 104},
    {VOLUME_GAIN_33dB, 108},
    {VOLUME_GAIN_36dB, 112},
```

```
    {VOLUME_GAIN_39dB, 116},
    {VOLUME_GAIN_42dB, 120},
    {VOLUME_GAIN_45dB, 124},
};
```

3.3.2.3 xradio_cld_vol_level[]

Audio PWM 不同播放音量等级与寄存器配置值映射关系数组，数组元素为 struct real_val_to_reg_val 结构体常量；如下代码段所示

```
static const struct real_val_to_reg_val xradio_cld_vol_level[] = {
    {VOLUME_LEVEL0, 0},
    {VOLUME_LEVEL1, 4},
    {VOLUME_LEVEL2, 8},
    {VOLUME_LEVEL3, 12},
    {VOLUME_LEVEL4, 16},
    {VOLUME_LEVEL5, 20},
    {VOLUME_LEVEL6, 24},
    {VOLUME_LEVEL7, 28},
    {VOLUME_LEVEL8, 32},
    {VOLUME_LEVEL9, 36},
    {VOLUME_LEVEL10, 40},
    {VOLUME_LEVEL11, 44},
    {VOLUME_LEVEL12, 48},
    {VOLUME_LEVEL13,  52},
    {VOLUME_LEVEL14,  56},
    {VOLUME_LEVEL15,  60},

    {VOLUME_LEVEL16,  64},

    {VOLUME_LEVEL17,  68},
    {VOLUME_LEVEL18,  72},
    {VOLUME_LEVEL19,  76},
    {VOLUME_LEVEL20, 80},
    {VOLUME_LEVEL21, 84},
    {VOLUME_LEVEL22, 88},
    {VOLUME_LEVEL23, 92},
    {VOLUME_LEVEL24, 96},
    {VOLUME_LEVEL25, 100},
    {VOLUME_LEVEL26, 104},
    {VOLUME_LEVEL27, 108},
    {VOLUME_LEVEL28, 112},
    {VOLUME_LEVEL29, 116},
    {VOLUME_LEVEL30, 120},
    {VOLUME_LEVEL31, 124},
```

```
};
```

3.3.3 基本读写寄存器接口

3.3.3.1 xradio_codec_reg_read

表 3-4 xradio_codec_reg_read 接口函数说明

信息项	说明
原型	static int xradio_codec_reg_read(uint32_t reg);
功能	读取 Audio PWM 寄存器值，相关寄存器的使用说明请查阅 Audio PWM 数据手册
参数	uint32_t reg 含义解释：需要读取的寄存器地址 使用说明：请参考 XR806 Audio PWM 数据手册寄存器列表
返回值	读取到的寄存器值

3.3.3.2 xradio_codec_reg_write

表 3-5 xradio_codec_reg_write 接口函数说明

信息项	说明
原型	static int xradio_codec_reg_write(uint32_t reg, uint32_t val);
功能	写入 Audio PWM 寄存器值，相关寄存器的使用说明请查阅 Audio PWM 数据手册
参数	Uint32_t reg 含义解释：需要写入的寄存器地址 使用说明：请参考 XR806 Audio PWM 数据手册寄存器列表 uint32_t val 含义解释：写入寄存器的实际值 使用说明：根据 XR806 Audio PWM 数据手册寄存器列表配置
返回值	HAL_OK：写入成功

3.3.3.3 xradio_codec_reg_update_bits

表 3-6 xradio_codec_reg_update_bits 接口函数说明

信息项	说明
原型	static int xradio_codec_reg_update_bits(uint32_t reg, uint32_t mask, uint32_t val);
功能	更新 Audio PWM 寄存器的某几 bit
参数	uint32_t reg 含义解释：需要更新的寄存器地址

信息项	说明
	使用说明：请参考 XR806 Audio PWM 数据手册寄存器列表 uint32_t mask 含义解释：需要更新的 bit mask 使用说明：根据 XR806 Audio PWM 数据手册寄存器列表配置 uint32_t val 含义解释：需要更新的 bit mask 对应值 使用说明：N/A
返回值	HAL_OK：更新成功

3.3.4 基本硬件配置接口

3.3.4.1 xradio_codec_reset

表 3-7 xradio_codec_reset 接口函数说明

信息项	说明
原型	static void xradio_codec_reset(void);
功能	Audio PWM 硬件复位，恢复到上电初始状态，包括 codec 相关的所有寄存器值
参数	无
返回值	无

3.3.4.2 xradio_codec_hw_common_init

表 3-8 xradio_codec_hw_common_init 接口函数说明

信息项	说明
原型	static void xradio_codec_hw_common_init(Audio_Stream_Dir dir);
功能	Audio PWM 公共硬件初始化配置，如时钟及数字部分 enable 等
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A
返回值	无

3.3.4.3 xradio_codec_hw_common_deinit

表 3-9 xradio_codec_hw_common_deinit 接口函数说明

信息项	说明
原型	static void xradio_codec_hw_common_deinit(Audio_Stream_Dir dir);
功能	Audio PWM 公共硬件反初始化配置，如时钟及数字部分 disable 等

信息项	说明
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A
返回值	无

3.3.5 DMA 配置接口

3.3.5.1 xradio_codec_dma_trigger

表 3-10 xradio_codec_dma_trigger 接口函数说明

信息项	说明
原型	static void xradio_codec_dma_trigger(Audio_Stream_Dir dir, bool enable);
功能	trigger DMA 以及配置 Audio PWM 与 DMA 相关的控制
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A bool enable 含义解释：DMA 启动/停止 trigger 控制 使用说明：N/A
返回值	无

3.3.5.2 xradio_codec_dma_threshold_check

表 3-11 xradio_codec_dma_threshold_check 接口函数说明

信息项	说明
原型	static int xradio_codec_dma_threshold_check(Audio_Stream_Dir dir);
功能	检查 DMA xrun 是否达到阈值，达到则停止 DMA 播放录音，未达到则直接返回
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A
返回值	HAL_OK：DMA xrun 未达到阈值 HAL_ERROR：DMA xrun 达到阈值，停止 DMA 播放/录音

3.3.5.3 xradio_codec_dma_half_callback

表 3-12 xradio_codec_dma_half_callback 接口函数说明

信息项	说明
原型	static void xradio_codec_dma_half_callback(void *arg);

信息项	说明
功能	DMA 半中断回调函数，更新半中断计数、释放占用信号量、检查 DMA xrun 次数、更新 DMA 当前读写指针
参数	void *arg 含义解释：DMA 当前占用的信号量 使用说明：N/A
返回值	无

3.3.5.4 xradio_codec_dma_end_callback

表 3-13 xradio_codec_dma_end_callback 接口函数说明

信息项	说明
原型	static void xradio_codec_dma_end_callback(void *arg);
功能	DMA 全中断回调函数，更新全中断计数、释放占用信号量、检查 DMA xrun 次数、更新 DMA 当前读写指针
参数	void *arg 含义解释：DMA 当前占用的信号量 使用说明：N/A
返回值	无

3.3.5.5 xradio_codec_dma_init

表 3-14 xradio_codec_dma_init 接口函数说明

信息项	说明
原型	static int xradio_codec_dma_init(Audio_Stream_Dir dir, DMA_Channel channel);
功能	初始化 DMA 配置参数
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A DMA_Channel channel 含义解释：DMA 通道 使用说明：N/A
返回值	HAL_OK：初始化成功 HAL_ERROR：初始化失败

3.3.6 ioctl 接口

3.3.6.1 xradio_codec_ioctl_pcm_write

表 3-15 xradio_codec_ioctl_pcm_write 接口函数说明

信息项	说明
原型	static int xradio_codec_ioctl_pcm_write(uint8_t *buf, uint32_t size);
功能	写入 PCM 数据
参数	uint8_t *buf 含义解释：指向写入 PCM 数据的 buff 使用说明：N/A uint32_t size 含义解释：写入 PCM 数据的大小 使用说明：N/A
返回值	大于 0：写入成功，返回已写入的数据量大小 小于等于 0：写入失败

3.3.7 codec_dai_ops 接口

Audio PWM 的 codec_dai_ops 接口封装如下代码段所示：

```
/*** codec dai ops ****/
static const struct codec_dai_ops xradio_codec_dai_ops = {
        .set_sysclk = xradio_dai_set_sysclk,
        .set_fmt        = xradio_dai_set_fmt,
        .set_volume = xradio_dai_set_volume,
        .set_route    = xradio_dai_set_route,
        .hw_params    = xradio_dai_hw_params,
        .hw_free        = xradio_dai_hw_free,
};
```

3.3.7.1 xradio_dai_set_sysclk

表 3-16 xradio_dai_set_sysclk 接口函数说明

信息项	说明
原型	static int xradio_dai_set_sysclk(Codec_Sysclk_Src sysclk_src, Codec_Pllclk_Src pllclk_src, uint32_t pll_freq_in, uint32_t sample_rate);
功能	配置 Audio PWM 系统时钟
参数	Codec_Sysclk_Src sysclk_src 含义解释：系统时钟源选择 使用说明：N/A Codec_Pllclk_Src pllclk_src

信息项	说明
	含义解释：PLL 时钟源选择 使用说明：N/A uint32_t pll_freq_in 含义解释：PLL 输入频率 使用说明：N/A uint32_t sample_rate 含义解释：采样率 使用说明：N/A
返回值	HAL_OK：配置成功 other：配置失败；

3.3.7.2 xradio_dai_set_fmt

表 3-17 xradio_dai_set_fmt 接口函数说明

信息项	说明
原型	static int xradio_dai_set_fmt(uint32_t fmt);
功能	未使用，此接口实现为空
参数	uint32_t fmt 含义解释：包括主从角色配置、I2S/LJ/RJ/PCM 格式配置、BCLK/LRCK 极性配置 使用说明：fmt 为 bit mask 组合配置，未使用
返回值	HAL_OK：配置成功

3.3.7.3 xradio_dai_set_volume

表 3-18 xradio_dai_set_volume 接口函数说明

信息项	说明
原型	static int xradio_dai_set_volume(Audio_Device device, uint16_t volume);
功能	配置 Audio PWM 对应通道的音量增益
参数	Audio_Device device 含义解释：AUDIO_Device 类型音频设备 使用说明：N/A uint16_t volume 含义解释：音量大小 使用说明：取值是枚举变量 Volume_Gain 或 Volume_Level。若选用 Volume_Gain 类型，则可取-45~45dB,3dB/Step；若选用 Volume_Level 类型，则可取 level0~level31

信息项	说明
返回值	HAL_OK：配置成功 other：配置失败；

3.3.7.4 xradio_dai_set_route

表 3-19 xradio_dai_set_route 接口函数说明

信息项	说明
原型	static int xradio_dai_set_route(Audio_Device device, Audio_Dev_State state);
功能	未使用，此接口实现为空
参数	Audio_Device device 含义解释：AUDIO_Device 类型音频设备 使用说明：N/A Audio_Dev_State state 含义解释：Audio_Dev_State 类型设备状态 使用说明：N/A
返回值	HAL_OK：配置成功 other：配置失败；

3.3.7.5 xradio_dai_hw_params

表 3-20 xradio_dai_hw_params 接口函数说明

信息项	说明
原型	static int xradio_dai_hw_params(Audio_Stream_Dir dir, struct pcm_config *pcm_cfg);
功能	配置 Audio PWM 的采样率、通道数、采样精度、buffer 大小、DMA 搬运数据宽度以及调用 xradio_codec_hw_common_init 接口进行公共硬件配置
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A struct pcm_config *pcm_cfg 含义解释：struct pcm_config 结构体类型指针 使用说明：N/A
返回值	HAL_OK：配置成功 other：配置失败；

3.3.7.6 xradio_dai_hw_free

表 3-21 xradio_dai_hw_free 接口函数说明

信息项	说明
原型	static int xradio_dai_hw_free(Audio_Stream_Dir dir);
功能	配置停止播放时调用 xradio_codec_hw_common_deinit 接口进行相关硬件清除操作，并调用 xradio_codec_reset 接口进行硬件复位操作
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A
返回值	HAL_OK：配置成功

3.3.8 codec_ops 接口

Audio PWM 的 codec_ops 接口封装如下代码段所示：

```
/*** codec ops ****/
static const struct codec_ops xradio_codec_ops = {
    .open    = xradio_codec_open,
    .close = xradio_codec_close,

    .reg_read    = xradio_codec_reg_read,
    .reg_write = xradio_codec_reg_write,

    .ioctl = xradio_codec_ioctl,
};
```

3.3.8.1 xradio_codec_open

表 3-22 xradio_codec_open 接口函数说明

信息项	说明
原型	static int xradio_codec_open(Audio_Stream_Dir dir);
功能	Audio PWM open 时进行相关配置操作，包括申请 DMA buffer、请求 DMA 通道、配置 DMA 参数等
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A
返回值	HAL_OK：配置成功 other：配置失败

3.3.8.2 xradio_codec_close

表 3-23 xradio_codec_close 接口函数说明

信息项	说明
原型	static int xradio_codec_close(Audio_Stream_Dir dir);
功能	Audio PWM close 时进行相关关闭操作，包括 DMA 反初始化、释放 DMA 通道、释放 DMA buffer 等
参数	Audio_Stream_Dir dir 含义解释：Audio_Stream_Dir 类型音频流方向 使用说明：N/A
返回值	HAL_OK：配置成功 other：配置失败

3.3.8.3 xradio_codec_ioctl

表 3-24 xradio_codec_ioctl 接口函数说明

信息项	说明
原型	static int xradio_codec_ioctl(uint32_t cmd, uint32_t cmd_param[], uint32_t cmd_param_len);
功能	对 Audio PWM 进行相关 ioctl 命令操作，如 pcm_write 等
参数	uint32_t cmd 含义解释：Codec_Ioctl_Cmd 类型命令 使用说明：N/A uint32_t cmd_param[] 含义解释：命令参数数组指针 使用说明：N/A uint32_t cmd_param_len 含义解释：命令参数数组长度 使用说明：N/A
返回值	HAL_INVALID：无效 ioctl 命令 other：相应命令的执行结果

3.3.9 codec_driver 接口

Audio PWM 的 codec_driver 接口封装如下代码段所示：

```
/*** codec driver ****/
static struct codec_driver xradio_internal_codec_drv = {
    .name = XRADIO_INTERNAL_CODEC_NAME,
    .codec_attr = XRADIO_CODEC_INTERNAL,
```

```
    .init = xradio_internal_codec_init,
    .deinit = xradio_internal_codec_deinit,

    .dai_ops = &xradio_codec_dai_ops,
    .codec_ops = &xradio_codec_ops,
};
```

3.3.9.1 xradio_internal_codec_init

表 3-25 xradio_internal_codec_init 接口函数说明

信息项	说明
原型	static int xradio_internal_codec_init(void);
功能	Audio PWM 模块全局硬件初始化，包括模块时钟 gating、复位的释放和 MCLK enable 等
参数	无
返回值	HAL_OK：配置成功

3.3.9.2 xradio_internal_codec_deinit

表 3-26 xradio_internal_codec_deinit 接口函数说明

信息项	说明
原型	static void xradio_internal_codec_deinit(void);
功能	Audio PWM 模块全局硬件反初始化，包括模块时钟 gating、复位的锁定和 MCLK disable 等
参数	无
返回值	无

3.3.10 注册接口

3.3.10.1 xradio_internal_codec_register

表 3-27 xradio_internal_codec_register 接口函数说明

信息项	说明
原型	HAL_Status xradio_internal_codec_register(void);
功能	Audio PWM 注册接口，包括申请私有数据内存空间、插入 Codec 链表等
参数	无
返回值	HAL_OK：注册成功 HAL_ERROR：注册失败

此接口不会在外部直接调用，会先在 HAL_SndCard 层封装后再统一给到外部调用，封装后的接口如下代码段所示：

```
HAL_Status HAL_SndCard_CodecRegisterInternal(void)
{
    return xradio_internal_codec_register();
}
```

3.3.10.2 xradio_internal_codec_unregister

表 3-28 xradio_internal_codec_unregister

信息项	说明
原型	HAL_Status xradio_internal_codec_unregister(void);
功能	Audio PWM 反注册接口，包括删除 Codec 链表项、释放私有数据内存空间等
参数	无
返回值	HAL_OK：反注册成功

此接口不会在外部直接调用，会先在 HAL_SndCard 层封装后再统一给到外部调用，封装后的接口如下代码段所示：

```
HAL_Status HAL_SndCard_CodecUnregisterInternal(void)
{
    return xradio_internal_codec_unregister();
}
```

4 示例说明

请参考文档《XR806 Audio 应用开发指南》示例。

著作权声明

商标声明

免责声明