XR806 Audio 应用开发指南

版本号：1.0

发布时间：2020-11-30

版本历史

版本	日期	责任人	版本描述
1.0	2020-11-30	AWA 1096	创建文档。

目录

版本历史 i
目录 ii
表格目录 2
1 前言 5
1.1 文档简介 5
1.2 目标读者 5
1.3 适用范围 5
1.4 文档约定 5
1.4.1 标志说明 5
2 概述 6
2.1 背景说明 6
2.2 规格特性 6
2.3 文件位置 8
3 应用说明 9
3.1 应用简述 9
3.2 配置说明 9
3.3 接口说明 9
3.3.1 CedarX 播放音频来源选择接口 10
3.3.1.1 CedarxStreamListInit 10
3.3.1.2 CedarxStreamRegisterHttps 10
3.3.1.3 CedarxStreamRegisterSsl 10
3.3.1.4 CedarxStreamRegisterFlash 10
3.3.1.5 CedarxStreamRegisterFile 11
3.3.1.6 CedarxStreamRegisterFifo 11
3.3.1.7 CedarxStreamRegisterHttp 11
3.3.1.8 CedarxStreamRegisterTcp 11
3.3.1.9 CedarxStreamRegisterCustomer 12
3.3.2 CedarX 播放音频格式选择接口 12
3.3.2.1 CedarxParserListInit 12
3.3.2.2 CedarxParserRegisterM3U 12
3.3.2.3 CedarxParserRegisterM4A 13
3.3.2.4 CedarxParserRegisterAAC 13
3.3.2.5 CedarxParserRegisterAMR 13
3.3.2.6 CedarxParserRegisterMP3 13
3.3.2.7 CedarxParserRegisterWAV 14

3.3.2.8 CedarxParserRegisterTS 14
3.3.3 CedarX 播放解码器选择接口 14
3.3.3.1 CedarxDecoderListInit 14
3.3.3.2 CedarxDecoderRegisterAAC 14
3.3.3.3 CedarxDecoderRegisterAMR 15
3.3.3.4 CedarxDecoderRegisterMP3 15
3.3.3.5 CedarxDecoderRegisterWAV 15
3.3.4 CedarX 播放接口 15
3.3.4.1 player_create 16
3.3.4.2 player_destroy 16
3.3.4.3 play 17
3.3.4.4 stop 17
3.3.4.5 pause 17
3.3.4.6 resume 18
3.3.4.7 seek 18
3.3.4.8 tell 18
3.3.4.9 size 19
3.3.4.10 setvol 19
3.3.4.11 getvol 19
3.3.4.12 mute 19
3.3.4.13 is_mute 20
3.3.4.14 control 20
3.3.4.15 set_callback 21
3.3.4.16 get_status 21
3.3.5 CedarX 录音音频地址选择接口 22
3.3.5.1 CedarxWriterListInit 22
3.3.5.2 CedarxWriterRegisterFile 22
3.3.5.3 CedarxWriterRegisterCallback 22
3.3.5.4 CedarxWriterRegisterCustomer 22
3.3.6 CedarX 录音音频格式选择接口 23
3.3.6.1 CedarxMuxerListInit 23
3.3.6.2 CedarxMuxerRegisterAmr 23
3.3.6.3 CedarxMuxerRegisterPcm 23
3.3.7 CedarX 录音编码器选择接口 24
3.3.7.1 CedarxEncoderListInit 24
3.3.7.2 CedarxEncoderRegisterAmr 24
3.3.7.3 CedarxEncoderRegisterPcm 24
3.3.8 CedarX 录音接口 24
3.3.8.1 recorder_create 24
3.3.8.2 recorder_destroy 25
3.3.8.3 start 25
3.3.8.4 stop 26

3.3.9 音频驱动数据流接口 26
3.3.9.1 snd_pcm_init 26
3.3.9.2 snd_pcm_deinit 27
3.3.9.3 snd_pcm_open 27
3.3.9.4 snd_pcm_close 28
3.3.9.5 snd_pcm_write 28
3.3.9.6 snd_pcm_read 29
3.3.9.7 snd_pcm_flush 29
3.3.10 音频驱动控制接口 29
3.3.10.1 audio_manager_init 29
3.3.10.2 audio_manager_handler 30
3.3.10.3 audio_manager_reg_read 30
3.3.10.4 audio_manager_reg_write 31
3.3.10.5 audio_manager_get_current_dev 31
4 示例说明 32
4.1 音频播放示例 32
4.1.1 示例简介 32
4.1.1.1 获取方式 32
4.1.1.2 准备工作 32
4.1.1.3 操作步骤 32
4.1.2 效果展示 32
4.1.3 关键实现 34
4.1.3.1 Cedarx 播放实现流程 34
4.1.3.2 音频驱动播放实现流程 35
4.2 音频录制示例 36
4.2.1 示例简介 36
4.2.1.1 获取方式 36
4.2.1.2 准备工作 36
4.2.1.3 操作步骤 36
4.2.2 效果展示 37
4.2.3 关键实现 38
4.2.3.1 Cedarx 录音实现流程 38
4.2.3.2 音频驱动录音实现流程 39

表格目录

表 2-1 CedarX 中间件的播放功能特性.....6
表 2-2 CedarX 中间件的录制功能特性.....7
表 2-3 音频驱动的播放功能特性.....7
表 2-4 音频驱动的录制功能特性.....7
表 2-5 CedarX 的文件位置.....8
表 2-6 Audio Driver 的文件位置.....8
表 3-1 音频播控模块配置列表.....9
表 3-2 音频模块常用接口简介.....9
表 3-3 CedarxStreamListInit 接口函数说明.....10
表 3-4 CedarxStreamRegisterHttps 接口函数说明.....10
表 3-5 CedarxStreamRegisterSsl 接口函数说明.....10
表 3-6 CedarxStreamRegisterFlash 接口函数说明.....10
表 3-7 CedarxStreamRegisterFile 接口函数说明.....11
表 3-8 CedarxStreamRegisterFifo 接口函数说明.....11
表 3-9 CedarxStreamRegisterHttp 接口函数说明.....11
表 3-10 CedarxStreamRegisterTcp 接口函数说明.....11
表 3-11 CedarxStreamRegisterCustomer 接口函数说明.....12
表 3-12 CedarxParserListInit 接口函数说明.....12
表 3-13 CedarxParserRegisterM3U 接口函数说明.....12
表 3-14 CedarxParserRegisterM4A 接口函数说明.....13
表 3-15 CedarxParserRegisterAAC 接口函数说明.....13
表 3-16 CedarxParserRegisterAMR 接口函数说明.....13
表 3-17 CedarxParserRegisterMP3 接口函数说明.....13
表 3-18 CedarxParserRegisterWAV 接口函数说明.....14
表 3-19 CedarxParserRegisterTS 接口函数说明.....14
表 3-20 CedarxDecoderListInit 接口函数说明.....14
表 3-21 CedarxDecoderRegisterAAC 接口函数说明.....14
表 3-22 CedarxDecoderRegisterAMR 接口函数说明.....15
表 3-23 CedarxDecoderRegisterMP3 接口函数说明.....15
表 3-24 CedarxDecoderRegisterWAV 接口函数说明.....15
表 3-25 player_create 接口函数说明.....16

表 3-26 player_destroy 接口函数说明 16
表 3-27 play 函数说明 17
表 3-28 stop 函数说明 17
表 3-29 pause 函数说明 17
表 3-30 resume 函数说明 18
表 3-31 seek 函数说明 18
表 3-32 tell 函数说明 18
表 3-33 size 函数说明 19
表 3-34 setvol 函数说明 19
表 3-35 getvol 函数说明 19
表 3-36 mute 函数说明 19
表 3-37 is_mute 函数说明 20
表 3-38 control 函数说明 20
表 3-39 set_callback 函数说明 21
表 3-40 get_status 函数说明 21
表 3-41 aplayer_states 枚举变量成员说明 21
表 3-42 CedarxWriterListInit 接口函数说明 22
表 3-43 CedarxWriterRegisterFile 接口函数说明 22
表 3-44 CedarxWriterRegisterCallback 接口函数说明 22
表 3-45 CedarxWriterRegisterCustomer 接口函数说明 22
表 3-46 CedarxMuxerListInit 接口函数说明 23
表 3-47 CedarxMuxerRegisterAmr 接口函数说明 23
表 3-48 CedarxMuxerRegisterPcm 接口函数说明 23
表 3-49 CedarxEncoderListInit 接口函数说明 24
表 3-50 CedarxEncoderRegisterAmr 接口函数说明 24
表 3-51 CedarxEncoderRegisterPcm 接口函数说明 24
表 3-52 recorder_create 接口函数说明 24
表 3-53 recorder_destroy 接口函数说明 25
表 3-54 start 函数说明 25
表 3-55 rec_cfg 结构体的成员说明 26
表 3-56 stop 函数说明 26
表 3-57 snd_pcm_init 接口函数说明 26
表 3-58 snd_pcm_deinit 接口函数说明 27

表 3-59 snd_pcm_open 接口函数说明.....27
表 3-60 struct pcm_config 结构体的成员说明.....27
表 3-61 snd_pcm_close 接口函数说明.....28
表 3-62 snd_pcm_write 接口函数说明.....28
表 3-63 snd_pcm_read 接口函数说明.....29
表 3-64 snd_pcm_flush 接口函数说明.....29
表 3-65 audio_manager_init 接口函数说明.....29
表 3-66 audio_manager_handler 接口函数说明.....30
表 3-67 audio_manager_reg_read 接口函数说明.....30
表 3-68 audio_manager_reg_write 接口函数说明.....31
表 3-69 audio_manager_get_current_dev 接口函数说明.....31

1 前言

1.1 文档简介

本文档介绍了 XR806 平台上音频播放和音频录制的方法。

1.2 目标读者

XR806 开发及维护人员。

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 支持音频播放和录制。对于纯音频数据的 PCM（Pulse Code Modulation，脉冲编码调制）音频，可以直接使用音频驱动进行播放和录制；对于需要进行编解码的音频，则可使用 CedarX 进行播放和录制。

CedarX 是一套多媒体中间件，主要负责音乐媒体的播放和录制。内部集成了流媒体处理、封装处理、编解码等复杂逻辑，对外提供简单的控制接口便于使用。它包含两个模块——XPlayer 播放、XRecord 录音，两个模块互相独立。XPlayer 负责把音频媒体转换成驱动可播放的 PCM 格式，驱动获取 PCM 数据后，会利用 CODEC 将 PCM 数据转换为模拟信号再由喇叭播放出来。XRecord 负责把驱动录下来的 PCM 数据转换成指定格式数据，存储到指定位置。

player_app 和 recorder_app 模块是对 XPlayer 播放模块和 XRecord 录音模块的封装，目的是使其播放/录音接口更加简单易用。

2.2 规格特性

表 2-1 CedarX 中间件的播放功能特性

规格类型	支持规格	规格描述	备注
输入方式	HTTP	播放 HTTP 音频	
	HTTPS	播放 HTTPS 音频	
	File	播放文件系统里的音频	
	flash	播放 flash 里的音频	
	音频数据流	播放音频数据流，该方式只支持 MP3、AMR、WAV 音频格式	
	自定义音频源	客户自己定义音频来源	
输入音频格式	MP3	播放 MP3 音频	
	AMR	播放 AMR 音频	
	M4A	播放 M4A 音频	
	M3U8	播放 M3U8 音频	
	WAV	播放 WAV 音频	
	AAC	播放 AAC 音频	
	TS	播放 TS 音频	
输出音频格式	PCM	输出的音频格式为 PCM 数据	
输出方式	SOUND CARD	输出音频数据到声卡进行播放	
	REVERB	输出音频数据到混响模块，供混响模块使用	

表 2-2 CedarX 中间件的录制功能特性

规格类型	支持规格	规格描述	备注
输入方式	SOUND CARD	从声卡获取音频数据进行录制	
输入音频格式	PCM	获取 PCM 数据进行录制	
输出音频格式	AMR	声道数：1 采样率：8000	
	PCM	声道数：1 采样率：8000、11025、12000、16000、22050、24000、32000	
输出方式	File	输出音频保存到文件系统	
	自定义音频流	客户自定义输出音频的保存方式	

表 2-3 音频驱动的播放功能特性

规格类型	支持规格	规格描述	备注
声道数	1	播放 1 声道的 PCM 数据	
	2	播放 2 声道的 PCM 数据	
采样率	8000	播放采样率 8000 的 PCM 数据	
	11025	播放采样率 11025 的 PCM 数据	
	12000	播放采样率 12000 的 PCM 数据	
	16000	播放采样率 16000 的 PCM 数据	
	22050	播放采样率 22050 的 PCM 数据	
	24000	播放采样率 24000 的 PCM 数据	
	44100	播放采样率 44100 的 PCM 数据	
	48000	播放采样率 48000 的 PCM 数据	

表 2-4 音频驱动的录制功能特性

规格类型	支持规格	规格描述	备注
声道数	1	录制 1 声道的 PCM 数据	
	2	录制 2 声道的 PCM 数据	
采样率	8000	录制采样率 8000 的 PCM 数据	
	11025	录制采样率 11025 的 PCM 数据	
	12000	录制采样率 12000 的 PCM 数据	
	16000	录制采样率 16000 的 PCM 数据	
	22050	录制采样率 22050 的 PCM 数据	

规格类型	支持规格	规格描述	备注
	24000	录制采样率 24000 的 PCM 数据	
	44100	录制采样率 44100 的 PCM 数据	
	48000	录制采样率 48000 的 PCM 数据	

2.3 文件位置

以 SDK 包为根目录，音频播控涉及到的主要文件位置如下。

表 2-5 CedarX 的文件位置

组件名	文件分类	文件位置
CedarX	库文件	sdk/lib/libcedarx.a
	源码文件	sdk/project/common/apps/cedarx sdk/project/common/apps/player_app.c sdk/project/common/apps/record_app.c
	头文件	sdk/include/cedarx sdk/project/common/apps/player_app.h sdk/project/common/apps/record_app.h
	示例工程	sdk/project/example/audio_play sdk/project/example/audio_record sdk/project/example/audio_play_and_record

表 2-6 Audio Driver 的文件位置

组件名	文件分类	文件位置
Audio Driver	源码文件	sdk/src/audio sdk/src/driver/chip/codec sdk/src/driver/chip/hal_snd_card.c
	头文件	sdk/include/audio/manager sdk/include/audio/pcm
	示例工程	sdk/project/example/audio_play sdk/project/example/audio_record sdk/project/example/audio_play_and_record

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

3 应用说明

3.1 应用简述

音频播控的代码已经内嵌到 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 音频播控模块配置列表

配置项	配置说明
使能接口	设置说明： 此项配置是否支持音频播控功能。 设置位置： /project/Kconfig 设置方式： 通过 make menuconfig，或修改 Kconfig，或修改编译工程的 defconfig 来选择使用的版本。如在 Kconfig 里选择默认支持音频播控功能： # xplayer config XPLAYER bool "Xplayer support" default y help xplayer.

3.3 接口说明

表 3-2 音频模块常用接口简介

接口名	简要介绍
CedarX 播放音频来源选择接口	本组接口用于选择支持的播放音频来源，属于外部接口。
CedarX 播放音频格式选择接口	本组接口用于选择支持的播放音频格式，属于外部接口。
CedarX 播放解码器选择接口	本组接口用于选择支持的播放解码器，属于外部接口。
CedarX 播放接口	本组接口用于播放音频，属于外部接口。
CedarX 录音音频地址选择接口	本组接口用于选择支持的录音存放地址，属于外部接口。
CedarX 录音音频格式选择接口	本组接口用于选择支持的录音音频格式，属于外部接口。
CedarX 录音编码器选择接口	本组接口用于选择支持的录音编码器，属于外部接口。
CedarX 录音接口	本组接口用于录制音频，属于外部接口。
音频驱动数据流接口	本组接口用于播放或录制音频，属于外部接口。

接口名	简要介绍
音频驱动控制接口	本组接口用于音频控制，属于外部接口。

3.3.1 CedarX 播放音频来源选择接口

3.3.1.1 CedarxStreamListInit

表 3-3 CedarxStreamListInit 接口函数说明

信息项	说明
原型	int CedarxStreamListInit(void);
功能	初始化音频来源支持列表。该函数必须被调用，且必须在注册 HTTPS、HTTP 等音频源前被调用。
参数	无
返回值	0：成功 -1：失败

3.3.1.2 CedarxStreamRegisterHttps

表 3-4 CedarxStreamRegisterHttps 接口函数说明

信息项	说明
原型	int CedarxStreamRegisterHttps(void);
功能	注册 HTTPS 音频源，使能支持 HTTPS 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.1.3 CedarxStreamRegisterSsl

表 3-5 CedarxStreamRegisterSsl 接口函数说明

信息项	说明
原型	int CedarxStreamRegisterSsl(void);
功能	注册 SSL 音频源，使能支持 HTTPS 音频的播放。SSL 是 HTTPS 的辅助音频源，播放 HTTPS 音频时需要该模块的支持。
参数	无
返回值	0：成功 -1：失败

3.3.1.4 CedarxStreamRegisterFlash

表 3-6 CedarxStreamRegisterFlash 接口函数说明

信息项	说明
原型	int CedarxStreamRegisterFlash(void);

信息项	说明
功能	注册 flash 音频源，使能支持 flash 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.1.5 CedarxStreamRegisterFile

表 3-7 CedarxStreamRegisterFile 接口函数说明

信息项	说明
原型	int CedarxStreamRegisterFile(void);
功能	注册 File 音频源，使能支持文件系统的音频播放。
参数	无
返回值	0：成功 -1：失败

3.3.1.6 CedarxStreamRegisterFifo

表 3-8 CedarxStreamRegisterFifo 接口函数说明

信息项	说明
原型	int CedarxStreamRegisterFifo(void);
功能	注册 fifo 音频源，使能支持音频数据流的播放。
参数	无
返回值	0：成功 -1：失败

3.3.1.7 CedarxStreamRegisterHttp

表 3-9 CedarxStreamRegisterHttp 接口函数说明

信息项	说明
原型	int CedarxStreamRegisterHttp(void);
功能	注册 HTTP 音频源，使能支持 HTTP 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.1.8 CedarxStreamRegisterTcp

表 3-10 CedarxStreamRegisterTcp 接口函数说明

信息项	说明
原型	int CedarxStreamRegisterTcp(void);

信息项	说明
功能	注册 TCP 音频源，使能支持 HTTP 音频的播放。TCP 是 HTTP 的辅助音频源，播放 HTTP 音频时需要该模块的支持。
参数	无
返回值	0：成功 -1：失败

3.3.1.9 CedarxStreamRegisterCustomer

表 3-11 CedarxStreamRegisterCustomer 接口函数说明

信息项	说明
原型	int CedarxStreamRegisterCustomer(void);
功能	注册 customer 音频源。customer 是用户自定义音频来源，用户需要定制音频来源时需要该模块的支持。
参数	无
返回值	0：成功 -1：失败

3.3.2 CedarX 播放音频格式选择接口

3.3.2.1 CedarxParserListInit

表 3-12 CedarxParserListInit 接口函数说明

信息项	说明
原型	int CedarxParserListInit(void);
功能	初始化音频格式支持列表。该函数必须被调用，且必须在注册 MP3、AMR 等播放格式前被调用。
参数	无
返回值	0：成功 -1：失败

3.3.2.2 CedarxParserRegisterM3U

表 3-13 CedarxParserRegisterM3U 接口函数说明

信息项	说明
原型	int CedarxParserRegisterM3U(void);
功能	注册 M3U8 音频格式，支持 M3U8 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.2.3 CedarxParserRegisterM4A

表 3-14 CedarxParserRegisterM4A 接口函数说明

信息项	说明
原型	int CedarxParserRegisterM4A(void);
功能	注册 M4A 音频格式，支持 M4A 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.2.4 CedarxParserRegisterAAC

表 3-15 CedarxParserRegisterAAC 接口函数说明

信息项	说明
原型	int CedarxParserRegisterAAC(void);
功能	注册 AAC 音频格式，支持 AAC 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.2.5 CedarxParserRegisterAMR

表 3-16 CedarxParserRegisterAMR 接口函数说明

信息项	说明
原型	int CedarxParserRegisterAMR(void);
功能	注册 AMR 音频格式，支持 AMR 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.2.6 CedarxParserRegisterMP3

表 3-17 CedarxParserRegisterMP3 接口函数说明

信息项	说明
原型	int CedarxParserRegisterMP3(void);
功能	注册 MP3 音频格式，支持 MP3 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.2.7 CedarxParserRegisterWAV

表 3-18 CedarxParserRegisterWAV 接口函数说明

信息项	说明
原型	int CedarxParserRegisterWAV(void);
功能	注册 WAV 音频格式，支持 WAV 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.2.8 CedarxParserRegisterTS

表 3-19 CedarxParserRegisterTS 接口函数说明

信息项	说明
原型	int CedarxParserRegisterTS(void);
功能	注册 TS 音频格式，支持 TS 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.3 CedarX 播放解码器选择接口

3.3.3.1 CedarxDecoderListInit

表 3-20 CedarxDecoderListInit 接口函数说明

信息项	说明
原型	int CedarxDecoderListInit(void);
功能	初始化解码器支持列表。该函数必须被调用，且必须在注册 AAC、MP3 等解码器前被调用。
参数	无
返回值	0：成功 -1：失败

3.3.3.2 CedarxDecoderRegisterAAC

表 3-21 CedarxDecoderRegisterAAC 接口函数说明

信息项	说明
原型	int CedarxDecoderRegisterAAC(void);
功能	注册 AAC 解码器。播放 M4A 和 AAC 音频时，需要使能支持 AAC 解码器模块。当 M3U8 里的播放列表存在 M4A 或 AAC 时，也需要使能支持 AAC 解码器模块。
参数	无

信息项	说明
返回值	0：成功 -1：失败

3.3.3.3 CedarxDecoderRegisterAMR

表 3-22 CedarxDecoderRegisterAMR 接口函数说明

信息项	说明
原型	int CedarxDecoderRegisterAMR(void);
功能	注册 AMR 解码器，支持 AMR 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.3.4 CedarxDecoderRegisterMP3

表 3-23 CedarxDecoderRegisterMP3 接口函数说明

信息项	说明
原型	int CedarxDecoderRegisterMP3(void);
功能	注册 MP3 解码器，支持 MP3 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.3.5 CedarxDecoderRegisterWAV

表 3-24 CedarxDecoderRegisterWAV 接口函数说明

信息项	说明
原型	int CedarxDecoderRegisterWAV(void);
功能	注册 WAV 解码器，支持 WAV 音频的播放。
参数	无
返回值	0：成功 -1：失败

3.3.4 CedarX 播放接口

音频播放需要先创建一个播放实体，然后利用播放实体进行音频播放、暂停、停止等操作。

3.3.4.1 player_create

表 3-25 player_create 接口函数说明

信息项	说明
原型	player_base * player_create();
功能	创建一个播放器实体。
参数	无
返回值	player_base 类型的结构体指针。该结构体包含了该播放器支持的操作方法。结构体的定义，以及各个操作方法的使用方法，请查看 player_base 结构体定义和下述各个接口说明。

player_base 结构体定义：

```
typedef struct player_base
{
    int (*play)(struct player_base *base, const char *url);
    int (*stop)(struct player_base *base);
    int (*pause)(struct player_base *base);
    int (*resume)(struct player_base *base);
    int (*seek)(struct player_base *base, int ms);
    int (*tell)(struct player_base *base);
    int (*size)(struct player_base *base);
    int (*setvol)(struct player_base *base, int vol);
    int (*getvol)(struct player_base *base);
    int (*mute)(struct player_base *base, bool is_mute);
    int (*is_mute)(struct player_base *base);
    int (*control)(struct player_base *base, player_cmd command, void *data);
    void (*set_callback)(struct player_base *base, app_player_callback cb, void *arg);
    aplayer_states (*get_status)(struct player_base *base);
} player_base;
```

3.3.4.2 player_destroy

表 3-26 player_destroy 接口函数说明

信息项	说明
原型	void player_destroy(player_base *base);
功能	销毁一个播放器实体。
参数	player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	无

3.3.4.3 play

表 3-27 play 函数说明

信息项	说明
定义	int (*play)(struct player_base *base, const char *url);
功能	player_base 结构体成员之一。用于播放一个 url 音频。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。 const char *url 含义解释：待播放的 url。 使用说明：赋值即可。
返回值	0：成功 -1：失败

3.3.4.4 stop

表 3-28 stop 函数说明

信息项	说明
定义	int (*stop)(struct player_base *base);
功能	player_base 结构体成员之一。用于停止播放。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	0：成功 -1：失败

3.3.4.5 pause

表 3-29 pause 函数说明

信息项	说明
定义	int (*pause)(struct player_base *base);
功能	player_base 结构体成员之一。用于暂停播放。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	0：成功 -1：失败

3.3.4.6 resume

表 3-30 resume 函数说明

信息项	说明
定义	int (*resume)(struct player_base *base);
功能	player_base 结构体成员之一。用于恢复播放。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	0：成功 -1：失败

3.3.4.7 seek

表 3-31 seek 函数说明

信息项	说明
定义	int (*seek)(struct player_base *base, int ms);
功能	player_base 结构体成员之一。用于跳转到指定位置进行播放。在指定了要播放的 url 后，就能使用 seek 函数进行跳转播放，如在播放中进行 seek，在暂停状态进行 seek。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。 int ms 含义解释：待跳转的位置 使用说明：赋值即可
返回值	0：成功 -1：失败

3.3.4.8 tell

表 3-32 tell 函数说明

信息项	说明
定义	int (*tell)(struct player_base *base);
功能	player_base 结构体成员之一。用于获取当前的播放进度。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	当前的播放进度

3.3.4.9 size

表 3-33 size 函数说明

信息项	说明
定义	int (*size)(struct player_base *base);
功能	player_base 结构体成员之一。用于获取当前播放音频的总时间长度。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	当前播放音频的总长度，单位为 ms。

3.3.4.10 setvol

表 3-34 setvol 函数说明

信息项	说明
定义	int (*setvol)(struct player_base *base, int vol);
功能	player_base 结构体成员之一。用于设置音量。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。 int vol 含义解释：音量值。 使用说明：赋值即可，取值范围 0~31。
返回值	0：成功 -1：失败

3.3.4.11 getvol

表 3-35 getvol 函数说明

信息项	说明
定义	int (*getvol)(struct player_base *base);
功能	player_base 结构体成员之一。用于获取当前音量。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	音量值

3.3.4.12 mute

表 3-36 mute 函数说明

信息项	说明
定义	int (*mute)(struct player_base *base, bool is_mute);

信息项	说明
功能	player_base 结构体成员之一。用于静音。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。 bool is_mute 含义解释：静音使能。 使用说明：赋值即可。1：静音；0：取消静音
返回值	0：成功 -1：失败

3.3.4.13 is_mute

表 3-37 is_mute 函数说明

信息项	说明
定义	int (*is_mute)(struct player_base *base);
功能	player_base 结构体成员之一。用于获取静音状态。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	返回是否静音

3.3.4.14 control

表 3-38 control 函数说明

信息项	说明
定义	int (*control)(struct player_base *base, player_cmd command, void *data);
功能	player_base 结构体成员之一。用于播放器控制。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。 player_cmd command 含义解释：执行的命令。 使用说明：赋值即可 void *data 含义解释：执行命令的参数。 使用说明：赋值即可
返回值	0：成功 -1：失败

3.3.4.15 set_callback

表 3-39 set_callback 函数说明

信息项	说明
定义	void (*set_callback)(struct player_base *base, app_player_callback cb, void *arg);
功能	player_base 结构体成员之一。用于设置回调函数。回调函数会反馈播放器的一些事件 event，如开始播放、播放结束、播放出错等。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。 app_player_callback cb 含义解释：状态回调函数。 使用说明：赋值即可 void *arg 含义解释：回调函数的参数。 使用说明：赋值即可
返回值	无

3.3.4.16 get_status

表 3-40 get_status 函数说明

信息项	说明
定义	aplayer_states (*get_status)(struct player_base *base);
功能	player_base 结构体成员之一。用于获取播放器状态。
参数	struct player_base *base 含义解释：播放器实体。 使用说明：player_create 创建的播放器实体。
返回值	播放器状态。播放器的状态，请参阅表 3-41 aplayer_states 枚举变量成员说明。

表 3-41 aplayer_states 枚举变量成员说明

成员项	说明
APLAYER_STATES_PAUSE	播放器处于暂停状态
APLAYER_STATES_PLAYING	播放器正在播放
APLAYER_STATES_PREPARING	播放器在准备阶段
APLAYER_STATES_PREPARED	播放器完成准备，可以启动播放
APLAYER_STATES_STOPPED	播放器已停止播放
APLAYER_STATES_INIT	播放器处于初始化阶段
APLAYER_STATES_COMPLETED	播放器已完成播放

成员项	说明
APLAYER_STATES_ERROR	播放器出错

3.3.5 CedarX 录音音频地址选择接口

3.3.5.1 CedarxWriterListInit

表 3-42 CedarxWriterListInit 接口函数说明

信息项	说明
原型	int CedarxWriterListInit(void);
功能	初始化音频地址支持列表。该函数必须被调用，且必须在注册 File、customer 等音频目的地址前被调用。
参数	无
返回值	0：成功 -1：失败

3.3.5.2 CedarxWriterRegisterFile

表 3-43 CedarxWriterRegisterFile 接口函数说明

信息项	说明
原型	int CedarxWriterRegisterFile(void);
功能	注册 File 音频地址，支持录音到文件系统。
参数	无
返回值	0：成功 -1：失败

3.3.5.3 CedarxWriterRegisterCallback

表 3-44 CedarxWriterRegisterCallback 接口函数说明

信息项	说明
原型	int CedarxWriterRegisterCallback(void);
功能	注册 callback 音频地址。callback 音频地址是自定义存放地址的一种实现方式，用户需实现自定义存放的函数实现。
参数	无
返回值	0：成功 -1：失败

3.3.5.4 CedarxWriterRegisterCustomer

表 3-45 CedarxWriterRegisterCustomer 接口函数说明

信息项	说明
原型	int CedarxWriterRegisterCustomer(void);

信息项	说明
功能	注册 customer 音频地址。customer 音频地址是自定义存放地址的另外一种实现方式，用户需实现自定义的 writer。推荐使用该方式实现自定义存放地址。
参数	无
返回值	0：成功 -1：失败

3.3.6 CedarX 录音音频格式选择接口

3.3.6.1 CedarxMuxerListInit

表 3-46 CedarxMuxerListInit 接口函数说明

信息项	说明
原型	int CedarxMuxerListInit(void);
功能	初始化音频格式支持列表。该函数必须被调用，且必须在注册 pcm、AMR 等音频格式前被调用。
参数	无
返回值	0：成功 -1：失败

3.3.6.2 CedarxMuxerRegisterAmr

表 3-47 CedarxMuxerRegisterAmr 接口函数说明

信息项	说明
原型	int CedarxMuxerRegisterAmr(void);
功能	注册 AMR 音频格式，支持录制 AMR 音频。
参数	无
返回值	0：成功 -1：失败

3.3.6.3 CedarxMuxerRegisterPcm

表 3-48 CedarxMuxerRegisterPcm 接口函数说明

信息项	说明
原型	int CedarxMuxerRegisterPcm(void);
功能	注册 pcm 音频格式，支持录制 PCM 音频。
参数	无
返回值	0：成功 -1：失败

3.3.7 CedarX 录音编码器选择接口

3.3.7.1 CedarxEncoderListInit

表 3-49 CedarxEncoderListInit 接口函数说明

信息项	说明
原型	int CedarxEncoderListInit(void);
功能	初始化编码器支持列表。该函数必须被调用，且必须在注册 AMR、PCM 等编码器前被调用。
参数	无
返回值	0：成功 -1：失败

3.3.7.2 CedarxEncoderRegisterAmr

表 3-50 CedarxEncoderRegisterAmr 接口函数说明

信息项	说明
原型	int CedarxEncoderRegisterAmr(void);
功能	注册 AMR 编码器，支持录制 AMR 音频。
参数	无
返回值	0：成功 -1：失败

3.3.7.3 CedarxEncoderRegisterPcm

表 3-51 CedarxEncoderRegisterPcm 接口函数说明

信息项	说明
原型	int CedarxEncoderRegisterPcm(void);
功能	注册 PCM 编码器，支持录制 PCM 音频。
参数	无
返回值	0：成功 -1：失败

3.3.8 CedarX 录音接口

音频录音需要先创建一个录音实体，然后利用录音实体进行录音。

3.3.8.1 recorder_create

表 3-52 recorder_create 接口函数说明

信息项	说明
原型	recorder_base *recorder_create();
功能	创建一个录音器实体。

信息项	说明
参数	无
返回值	recorder_base 类型的结构体指针。该结构体包含了该录音器支持的操作方法。结构体的定义，以及各个操作方法的使用方法，请查看 struct recorder_base 结构体定义和下述各个接口说明。

struct recorder_base 结构体定义：

```
struct recorder_base
{
    int (*start)(recorder_base *base, const char *url, const rec_cfg *cfg);
    int (*stop)(recorder_base *base);
};
```

3.3.8.2 recorder_destroy

表 3-53 recorder_destroy 接口函数说明

信息项	说明
原型	int recorder_destroy(recorder_base *base);
功能	销毁一个录音器实体。
参数	recorder_base *base 含义解释：录音器实体。 使用说明：recorder_create 创建的录音器实体。
返回值	0：成功 -1：失败

3.3.8.3 start

表 3-54 start 函数说明

信息项	说明
定义	int (*start)(recorder_base *base, const char *url, const rec_cfg *cfg);
功能	struct recorder_base 结构体成员之一。用于启动录音。
参数	recorder_base *base 含义解释：录音器实体。 使用说明：recorder_create 创建的录音器实体。 const rec_cfg *cfg 含义解释：rec_cfg 结构体指针，此结构体存放录音的配置信息。 使用说明：rec_cfg 结构体的成员说明，请参阅表 3-51 rec_cfg 结构体的成员说明。
返回值	0：成功 -1：失败

表 3-55 rec_cfg 结构体的成员说明

成员项	说明
XRECODER_AUDIO_ENCODE_TYPE type;	含义解释：录音类型。 使用说明：赋值即可，选择录音为 ARM 还是 PCM。
int sample_rate;	含义解释：采样率。 使用说明：赋值即可。如果录取 AMR，则只支持 8000 采样率。
int chan_num;	含义解释：声道数。 使用说明：赋值即可。如果录取 AMR，则只支持 1 声道。
int bitrate;	含义解释：码率。 使用说明：赋值即可，目前只支持 12200。
int sampler_bits;	含义解释：单个采样点的位深。 使用说明：赋值即可，目前只支持 16bits。

3.3.8.4 stop

表 3-56 stop 函数说明

信息项	说明
定义	int (*stop)(recorder_base *base);
功能	struct recorder_base 结构体成员之一。用于停止录音。
参数	recorder_base *base 含义解释：录音器实体。 使用说明：recorder_create 创建的录音器实体。
返回值	0：成功 -1：失败

3.3.9 音频驱动数据流接口

3.3.9.1 snd_pcm_init

表 3-57 snd_pcm_init 接口函数说明

信息项	说明
原型	int snd_pcm_init(void);
功能	初始化音频驱动数据流模块，系统上电初始化时调用。
参数	无
返回值	0：成功 -1：失败

3.3.9.2 snd_pcm_deinit

表 3-58 snd_pcm_deinit 接口函数说明

信息项	说明
原型	int snd_pcm_deinit(void);
功能	反初始化音频驱动数据流模块。
参数	无
返回值	0：成功 -1：失败

3.3.9.3 snd_pcm_open

表 3-59 snd_pcm_open 接口函数说明

信息项	说明
原型	int snd_pcm_open(Snd_Card_Num card_num, Audio_Stream_Dir stream_dir, struct pcm_config *pcm_cfg);
功能	打开一个 PCM 流。
参数	Snd_Card_Num card_num 含义解释：声卡号。 使用说明：赋值即可。 Audio_Stream_Dir stream_dir 含义解释：pcm 数据流方向，用来决定是录音还是播放。 使用说明：赋值即可。 struct pcm_config *pcm_cfg 含义解释：struct pcm_config 结构体指针，此结构体存放录音或播放的配置信息。 使用说明：struct pcm_config 结构体的成员说明，请参阅表 3-58 struct pcm_config 结构体的成员说明。
返回值	0：成功 -1：失败

表 3-60 struct pcm_config 结构体的成员说明

成员项	说明
uint32_t rate;	含义解释：采样率 使用说明：赋值即可
uint32_t channels;	含义解释：声道数 使用说明：赋值即可
uint32_t period_size;	含义解释：音频驱动的 buffer 大小配置

成员项	说明
	使用说明：赋值即可。采样率越高需要的 buffer 越大；采样率越低需要的 buffer 越小。该值配置为 1024 可满足大部分需求。
uint32_t period_count;	含义解释：音频驱动的 buffer 个数。 使用说明：赋值即可，固定设置为 2
enum pcm_format format;	含义解释：单个采样点的位深。 使用说明：赋值即可

3.3.9.4 snd_pcm_close

表 3-61 snd_pcm_close 接口函数说明

信息项	说明
原型	int snd_pcm_close(Snd_Card_Num card_num, Audio_Stream_Dir stream_dir);
功能	关闭一个 PCM 流。
参数	Snd_Card_Num card_num 含义解释：要关闭的声卡号。 使用说明：赋值即可。 Audio_Stream_Dir stream_dir 含义解释：pcm 数据流方向，用来决定是关闭录音还是关闭播放。 使用说明：赋值即可。
返回值	0：成功 -1：失败

3.3.9.5 snd_pcm_write

表 3-62 snd_pcm_write 接口函数说明

信息项	说明
原型	int snd_pcm_write(Snd_Card_Num card_num, void *data, uint32_t count);
功能	往对应声卡写入播放 pcm 数据。
参数	Snd_Card_Num card_num 含义解释：要操作的声卡号。 使用说明：赋值即可。 void *data 含义解释：要写的数据。 使用说明：赋值即可。 uint32_t count 含义解释：写入数据的长度。 使用说明：赋值即可。

信息项	说明
返回值	返回已写入的数据量长度

3.3.9.6 snd_pcm_read

表 3-63 snd_pcm_read 接口函数说明

信息项	说明
原型	int snd_pcm_read(Snd_Card_Num card_num, void *data, uint32_t count);
功能	读取对应声卡录音 pcm 数据。
参数	Snd_Card_Num card_num 含义解释：要操作的声卡号。 使用说明：赋值即可。 void *data 含义解释：缓存 buffer。 使用说明：赋值即可。 uint32_t count 含义解释：希望读取的数据的长度。 使用说明：赋值即可。
返回值	返回读取到的数据量长度

3.3.9.7 snd_pcm_flush

表 3-64 snd_pcm_flush 接口函数说明

信息项	说明
原型	int snd_pcm_flush(Snd_Card_Num card_num);
功能	将音频驱动缓存区里的 pcm 数据冲刷到硬件，应用层播放完毕时调用。
参数	Snd_Card_Num card_num 含义解释：要操作的声卡号。 使用说明：赋值即可。
返回值	0：成功 -1：失败

3.3.10 音频驱动控制接口

3.3.10.1 audio_manager_init

表 3-65 audio_manager_init 接口函数说明

信息项	说明
原型	int audio_manager_init(void);
功能	初始化音频驱动控制模块，系统上电初始化时调用。
参数	无

信息项	说明
返回值	0：成功 -1：失败

3.3.10.2 audio_manager_handler

表 3-66 audio_manager_handler 接口函数说明

信息项	说明
原型	int audio_manager_handler(Snd_Card_Num card_num, Audio_Manager_Cmd cmd, Audio_Device dev, uint32_t param);
功能	AUDIO Manager 层音频控制接口，如设置音量、路径、Mute/Unmute 等。
参数	Snd_Card_Num card_num 含义解释：要操作的声卡号。 使用说明：赋值即可。 Audio_Manager_Cmd cmd 含义解释：控制命令。 使用说明：赋值即可。 Audio_Device dev 含义解释：要操作的音频设备。 使用说明：赋值即可。 uint32_t param 含义解释：命令参数。 使用说明：赋值即可。
返回值	0：成功 -1：失败

3.3.10.3 audio_manager_reg_read

表 3-67 audio_manager_reg_read 接口函数说明

信息项	说明
原型	int audio_manager_reg_read(Snd_Card_Num card_num, uint32_t reg);
功能	读取对应声卡的 codec 寄存器值。
参数	Snd_Card_Num card_num 含义解释：要操作的声卡号。 使用说明：赋值即可。 uint32_t reg 含义解释：读取的寄存器地址。 使用说明：赋值即可。
返回值	读取到的寄存器值

3.3.10.4 audio_manager_reg_write

表 3-68 audio_manager_reg_write 接口函数说明

信息项	说明
原型	int audio_manager_reg_write(Snd_Card_Num card_num, uint32_t reg, uint32_t val);
功能	往对应声卡的 codec 写入对应寄存器值。
参数	Snd_Card_Num card_num 含义解释：要操作的声卡号。 使用说明：赋值即可。 uint32_t reg 含义解释：操作的寄存器地址。 使用说明：赋值即可。 uint32_t val 含义解释：要写入的寄存器值。 使用说明：赋值即可。
返回值	0：成功 -1：失败

3.3.10.5 audio_manager_get_current_dev

表 3-69 audio_manager_get_current_dev 接口函数说明

信息项	说明
原型	int audio_manager_get_current_dev(Snd_Card_Num card_num);
功能	获取当前播放录音设备。
参数	Snd_Card_Num card_num 含义解释：要操作的声卡号。 使用说明：赋值即可。
返回值	当前播放录音设备的 bit mask

4 示例说明

Audio 接口用于进行音频播放和音频录制，下面以音频播放和音频录制为例，简要说明 Audio 接口的使用。

4.1 音频播放示例

4.1.1 示例简介

音频播放示例的演示的目的是简要介绍 Cedarx 音频播放和音频驱动相关接口的基本使用方法，演示的功能为播放不同地址的音频。

4.1.1.1 获取方式

音频播放示例有示例工程代码，位于 XR806 SDK 的/project/example/audio_play 目录。

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

4.1.1.2 准备工作

音频播放示例工程的硬件准备有如下。

1. 评估板：运行示例工程代码。
2. 串口线：连接评估板的 Uart0 插针，用于 console 控制台的输入输出。
3. PC 机：用于镜像烧录和 console 控制的输入输出。
4. 喇叭：用于播放

音频播放的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》。

4.1.1.3 操作步骤

1. 存放一首音频到文件系统 music 目录，并命名该音频为”1.mp3”。可由 menuconfig 选择使用的文件系统，以及文件系统的起始地址、文件系统大小。该工程默认的起始地址为 1.5M，大小为 512k。即默认需要将包含音频 1.mp3 的文件系统烧写到 1.5M flash 位置，大小不超过 512k。
2. 存放一段单声道、16k 采样率的 PCM 音频到文件系统 music 目录，并命名该音频为”16000_1.pcm”。
3. 开发板完成烧写，复位，通过串口命令连接网络后，示例代码自动运行。

4.1.2 效果展示

在评估板中运行音频播放示例程序后，能在喇叭听到播放的音频，也能在控制台中会打印出音频播放的现状，如下所示。

```
5fae4f24
[PTHREAD INFO] pthread_create line 478, thread: 0x20e1c0 create success!
[PTHREAD INFO] pthread_create line 478, thread: 0x20e6a8 create success!
xplayer thread dealing with msg.messageId: 0x103
```

```
player create success.
you can use it to play, pause, resume, set volume and so on.
player set volume to 16. valid volume value is from 0~31
[XRADIO_INTERNAL_CODEC] CLD set volume Level-[16]
===try to play media in file system===
[PLAYER_INFO] <player_seturl : 445> request to play : file://data/music/1.mp3
xplayer thread dealing with msg.messageId: 0x101
xplayer thread dealing with msg.messageId: 0x104
prepare finish
parser create success
parser create success
mp3 parser init success
[PTHREAD INFO] pthread_create line 478, thread: 0x20f270 create success!
[PTHREAD INFO] pthread_create line 478, thread: 0x211708 create success!
[PLAYER_INFO] <player_handler_preprocess : 187> event: 1
media is prepared. play media now.
you can use player to seek, get duration(by size), get current position(by tell) from now on.
xplayer thread dealing with msg.messageId: 0x105
[XRADIO_INTERNAL_CODEC] CLD set volume Level-[16]
music pause now, last for 10s
xplayer thread dealing with msg.messageId: 0x107
music resume now.
xplayer thread dealing with msg.messageId: 0x105
[XRADIO_INTERNAL_CODEC] CLD set volume Level-[16]
[PLAYER_INFO] <player_size : 489> size to 20145 ms.
[PLAYER_INFO] <player_tell : 473> tell to 10111 ms.
this song is 20145ms in length, and it has play 10111ms
we will seek to the half of this song now.
xplayer thread dealing with msg.messageId: 0x10a
[XRADIO_INTERNAL_CODEC] CLD set volume Level-[16]
[PLAYER_INFO] <player_seek : 420> seek to 10072 ms.
[PLAYER_INFO] <player_tell : 473> tell to 10072 ms.
now this song has play 10072ms
[PLAYER_INFO] <player_callback : 322> seek ok.
[PLAYER_INFO] <player_callback : 330> cedarx cb complete.
[PLAYER_INFO] <player_callback : 317> playback complete.
[PLAYER_INFO] <player_callback : 330> cedarx cb complete.
[PLAYER_INFO] <player_handler_preprocess : 187> event: 8
xplayer thread dealing with msg.messageId: 0x108
media play is completed
```

4.1.3 关键实现

4.1.3.1 Cedarx 播放实现流程

第一步：确定需要支持的播放功能特性。

在用户的应用代码中增加 platform_cedarx_init 函数实现。可以参考 platform_init.c 文件中 platform_cedarx_init 的实现。需要包含头文件"cedarx/cedarx.h"。

```
void platform_cedarx_init(void)
{
    /* for media player */
    CedarxStreamListInit();
#if PRJCONF_NET_EN
    CedarxStreamRegisterHttps();
    CedarxStreamRegisterSsl();
    CedarxThreadStackSizeSet(DEMUX_THREAD, 8 * 1024);
    CedarxStreamRegisterHttp();
    CedarxStreamRegisterTcp();
#endif
    CedarxStreamRegisterFlash();
#ifdef CONFIG_FILESYSTEMS
    CedarxStreamRegisterFile();
#endif
    CedarxStreamRegisterFifo();
    CedarxStreamRegisterCustomer();

    CedarxParserListInit();
    CedarxParserRegisterM3U();
    CedarxParserRegisterM4A();
    CedarxParserRegisterAAC();
    CedarxParserRegisterAMR();
    CedarxParserRegisterMP3();
    CedarxParserRegisterWAV();
    CedarxParserRegisterTS();

    CedarxDecoderListInit();
    CedarxDecoderRegisterAAC();
    CedarxDecoderRegisterAMR();
    CedarxDecoderRegisterMP3();
    CedarxDecoderRegisterWAV();

    SoundStreamListInit();
    SoundStreamRegisterCard();
    SoundStreamRegisterReverb();
}
```

说明

platform_init.c 文件中的 platform_cedarx_init 函数被定义为__weak 类型，因此用户可以通过在应用代码中定义同名函数将其覆盖。不建议用户直接修改此处的定义。

第二步：创建播放器。

```
static player_base *player;
player = player_create();
```

第三步：利用播放器进行播放、暂停、重启播放、跳播等。

```
player->set_callback(player, player_demo_callback, NULL);
ret = player->play(player, "file://data/music/1.mp3");
if (ret != 0) {
    printf("music play fail.\n");
    return -1;
}

OS_Sleep(15);
printf("music pause now, last for 10s\n");
player->pause(player);
OS_Sleep(10);

printf("music resume now.\n");
player->resume(player);

OS_Sleep(20);
duration = player->size(player);
position = player->tell(player);
printf("this song is %dms in length, and it has play %dms\n", duration, position);

printf("we will seek to the half of this song now.\n");
player->seek(player, duration / 2);

position = player->tell(player);
printf("now this song has play %dms\n", position);
```

第四步：播放结束后，停止播放，恢复播放器到初始状态。

```
player->stop(player);
```

4.1.3.2 音频驱动播放实现流程

第一步：根据要播放的 PCM 流打开声卡。

```
struct pcm_config config;

config.channels = 1;
config.format = PCM_FORMAT_S16_LE;
config.period_count = 2;
config.period_size = 1024;
config.rate = 16000;
ret = snd_pcm_open(AUDIO_SND_CARD_DEFAULT, PCM_OUT, &config);
```

第二步：写入 PCM 数据（可多次写入）。

```
snd_pcm_write(AUDIO_SND_CARD_DEFAULT, pcm_buf, act_read);
```

第三步：冲刷缓存在 cache 里面的 PCM 数据。

```
snd_pcm_flush(AUDIO_SND_CARD_DEFAULT);
```

第四步：关闭声卡。

```
snd_pcm_close(AUDIO_SND_CARD_DEFAULT, PCM_OUT);
```

4.2 音频录制示例

4.2.1 示例简介

音频录制示例的演示的目的是简要介绍 Cedarx 音频录制和音频驱动相关接口的基本使用方法，演示的功能为录制不同类型的音频。

4.2.1.1 获取方式

音频播放示例有示例工程代码，位于 XR806 SDK 的/project/example/audio_record 目录。

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

4.2.1.2 准备工作

音频录制示例工程的硬件准备有如下。

5. 评估板：需要有外置 codec。audio_record 示例工程默认使用外置 AC101 进行录音
6. 串口线：连接评估板的 Uart0 插针，用于 console 控制台的输入输出。
7. PC 机：用于镜像烧录和 console 控制的输入输出。

音频录制的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》

4.2.1.3 操作步骤

开发板完成烧写，复位，通过串口命令连接网络后，示例代码自动运行。

4.2.2 效果展示

在评估板中运行音频录制示例程序后，在控制台中会打印出录音的各个步骤和进度。完成录音后，可以在文件系统里拿到录制的音频文件。如下所示。

```
audio record start.
[AC101_CODEC] AMIC set volume Level-[3]
start to use cedarx to record amr/pcm
[PTHREAD INFO] pthread_create line 478, thread: 0x20eee0 create success!
[PTHREAD INFO] pthread_create line 478, thread: 0x20fa88 create success!
===start record amr now, last for 15s===
[PTHREAD INFO] pthread_create line 478, thread: 0x212118 create success!
[PTHREAD INFO] pthread_create line 478, thread: 0x218ae0 create success!
[AC101_CODEC] AMIC set volume Level-[3]
[AC101_CODEC] LINEIN set volume Level-[4]
[AC101_CODEC] Route(cap): amic Enable
[RECORDER_INFO] <record_start : 82> record start
[AC101_CODEC] Route(cap): amic Disable
[AC101_CODEC] Route(cap): linein Disable
[AC101_CODEC] Route(cap): dmic Disable
record amr over.
===start record pcm now, last for 15s===
[PTHREAD INFO] pthread_create line 478, thread: 0x2172e8 create success!
[PTHREAD INFO] pthread_create line 478, thread: 0x2113e0 create success!
[AC101_CODEC] AMIC set volume Level-[3]
[AC101_CODEC] LINEIN set volume Level-[4]
[AC101_CODEC] Route(cap): amic Enable
[RECORDER_INFO] <record_start : 82> record start
[AC101_CODEC] Route(cap): amic Disable
[AC101_CODEC] Route(cap): linein Disable
[AC101_CODEC] Route(cap): dmic Disable
record pcm over.
===start record amr by customer writer now, last for 15s===
[PTHREAD INFO] pthread_create line 478, thread: 0x217360 create success!
[PTHREAD INFO] pthread_create line 478, thread: 0x210998 create success!
[AC101_CODEC] AMIC set volume Level-[3]
[AC101_CODEC] LINEIN set volume Level-[4]
[AC101_CODEC] Route(cap): amic Enable
[RECORDER_INFO] <record_start : 82> record start
[AC101_CODEC] Route(cap): amic Disable
[AC101_CODEC] Route(cap): linein Disable
[AC101_CODEC] Route(cap): dmic Disable
record amr over.
start to use audio driver to record pcm
[AC101_CODEC] AMIC set volume Level-[3]
```

```
[AC101_CODEC] LINEIN set volume Level-[4]
[AC101_CODEC] Route(cap): amic Enable
===start record pcm by audio driver, last for 10000ms===
record pcm over.
[AC101_CODEC] Route(cap): amic Disable
[AC101_CODEC] Route(cap): linein Disable
[AC101_CODEC] Route(cap): dmic Disable
audio record over.
```

4.2.3 关键实现

4.2.3.1 Cedarx 录音实现流程

第一步：确定需要支持的录音功能特性。

在用户的应用代码中增加 platform_cedarx_init 函数实现。可以参考 platform_init.c 文件中 platform_cedarx_init 的实现。需要包含头文件"cedarx/cedarx.h"。

```
void platform_cedarx_init(void)
{
    CedarxWriterListInit();
#ifdef CONFIG_FILESYSTEMS
    CedarxWriterRegisterFile();
#endif
    CedarxWriterRegisterCallback();
    CedarxWriterRegisterCustomer();

    CedarxMuxerListInit();
    CedarxMuxerRegisterAmr();
    CedarxMuxerRegisterPcm();

    CedarxEncoderListInit();
    CedarxEncoderRegisterAmr();
    CedarxEncoderRegisterPcm();
}
```

说明

platform_init.c 文件中的 platform_cedarx_init 函数被定义为__weak 类型，因此用户可以通过在应用代码中定义同名函数将其覆盖。不建议用户直接修改此处的定义。

第二步：创建录音器。

```
recorder_base *recorder;
recorder = recorder_create();
```

第三步：利用录音器进行录音。

```
/* record a 15s amr media */
cfg.type = XRECODER_AUDIO_ENCODE_AMR_TYPE;
printf("===start record amr now, last for 15s===\n");
recorder->start(recorder, "file://data/record/1.amr", &cfg);
OS_Sleep(15);
recorder->stop(recorder);
printf("record amr over.\n");

/* record a 15s pcm media */
cfg.type = XRECODER_AUDIO_ENCODE_PCM_TYPE;
cfg.sample_rate = 8000;
cfg.chan_num = 1;
cfg.bitrate = 12200;
cfg.sampler_bits = 16;
printf("===start record pcm now, last for 15s===\n");
recorder->start(recorder, "file://data/record/1.pcm", &cfg);
OS_Sleep(15);
recorder->stop(recorder);
printf("record pcm over.\n");
```

第四步：销毁录音器。

```
recorder_destroy(recorder);
```

4.2.3.2 音频驱动录音实现流程

第一步：根据要录制的 PCM 流打开声卡。

```
struct pcm_config config;

config.channels = 1;
config.format = PCM_FORMAT_S16_LE;
config.period_count = 2;
config.period_size = 1024;
config.rate = 8000;
ret = snd_pcm_open(AUDIO_CARD, PCM_IN, &config);
```

第二步：读取 PCM 数据（可多次读取）。

```
snd_pcm_read(AUDIO_CARD, data, len);
```

第三步：关闭声卡。

```
snd_pcm_close(AUDIO_CARD, PCM_IN);
```

著作权声明

商标声明

免责声明