SNC86xx SDK 二次开发说明

V1.1

History

版本	发布时间	版本说明	作者	核准人
Draft	2022-4-10	1st version release	白蓉	
1.0	2022-4-27	1st version for official release	白蓉	
1.1	2022-4-30	增加 SDK 及 AudioManager 架构说明	白蓉	

目录

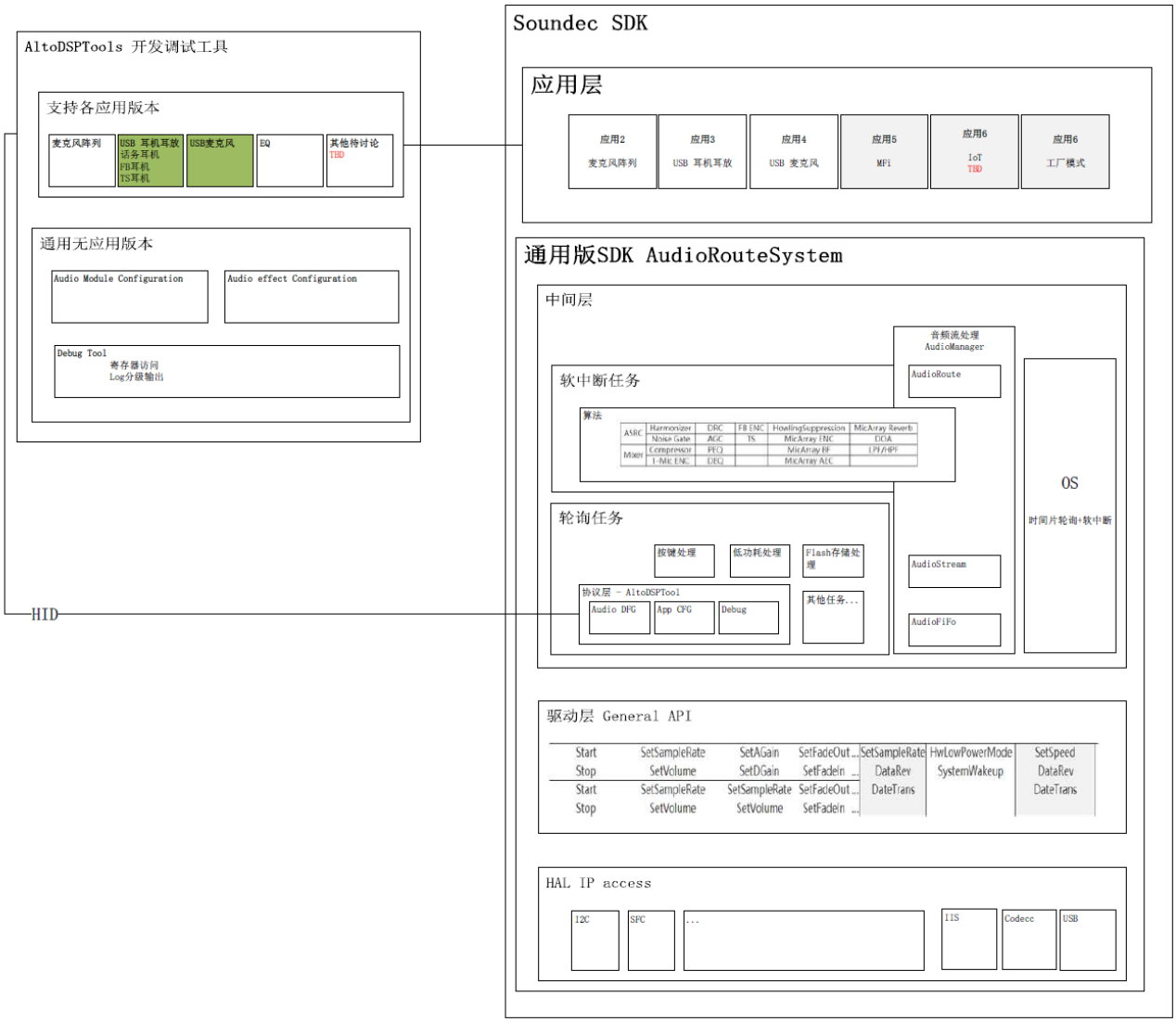
1 概述

1.1 Soundec32CxxSDK 简介

Soundec32CxxSDK 是深圳市九音科技有限公司提供的基于自研的 SNC86xx 系列 DSP 芯片的开发平台。

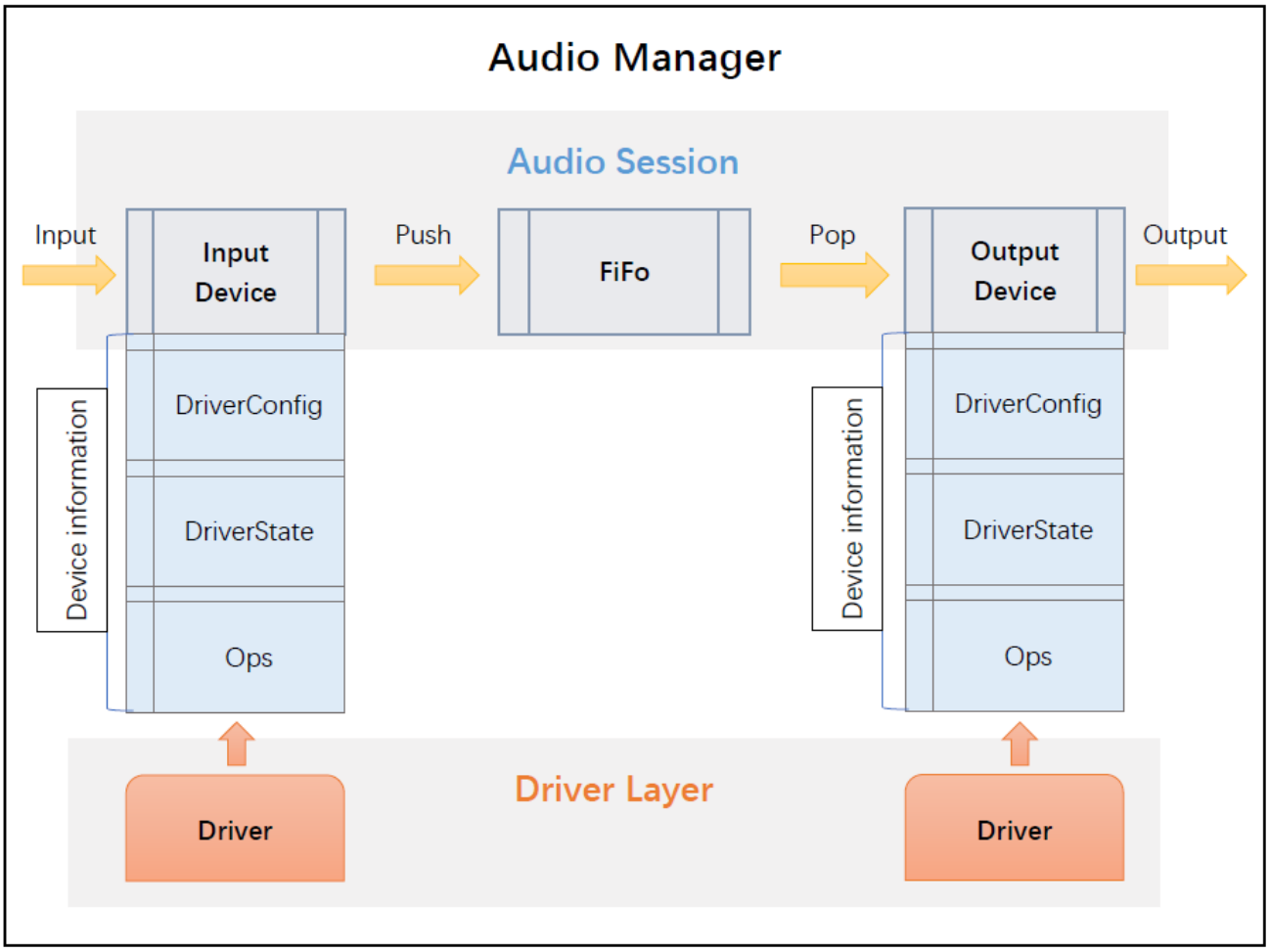
1.2.1 Soundec32CxxSDK 架构

Soundec32CxxSDK 提供完整的驱动层及调用接口，提供外设单元的调用例程，提供 DSP 调试工具用于方便的设置音频单元参数。基于 Soundec32CxxSDK，可以方便的支持用户扩展所需应用，并为第三方音频处理算法预留处理接口，方便算法的导入。



1.2.2 AudioManager 架构

Soundec32CxxSDK 以音频处理为主要架构，根据应用的音频通路需求，用户只需要新建和配置音频会话描述文件（audio_session_desc_xxx.c），系统即可通过 AudioManager 自动加载对应的音频 Device，从而完成音频模块的便携式配置，实现多入多出音频流通路的自由组合。

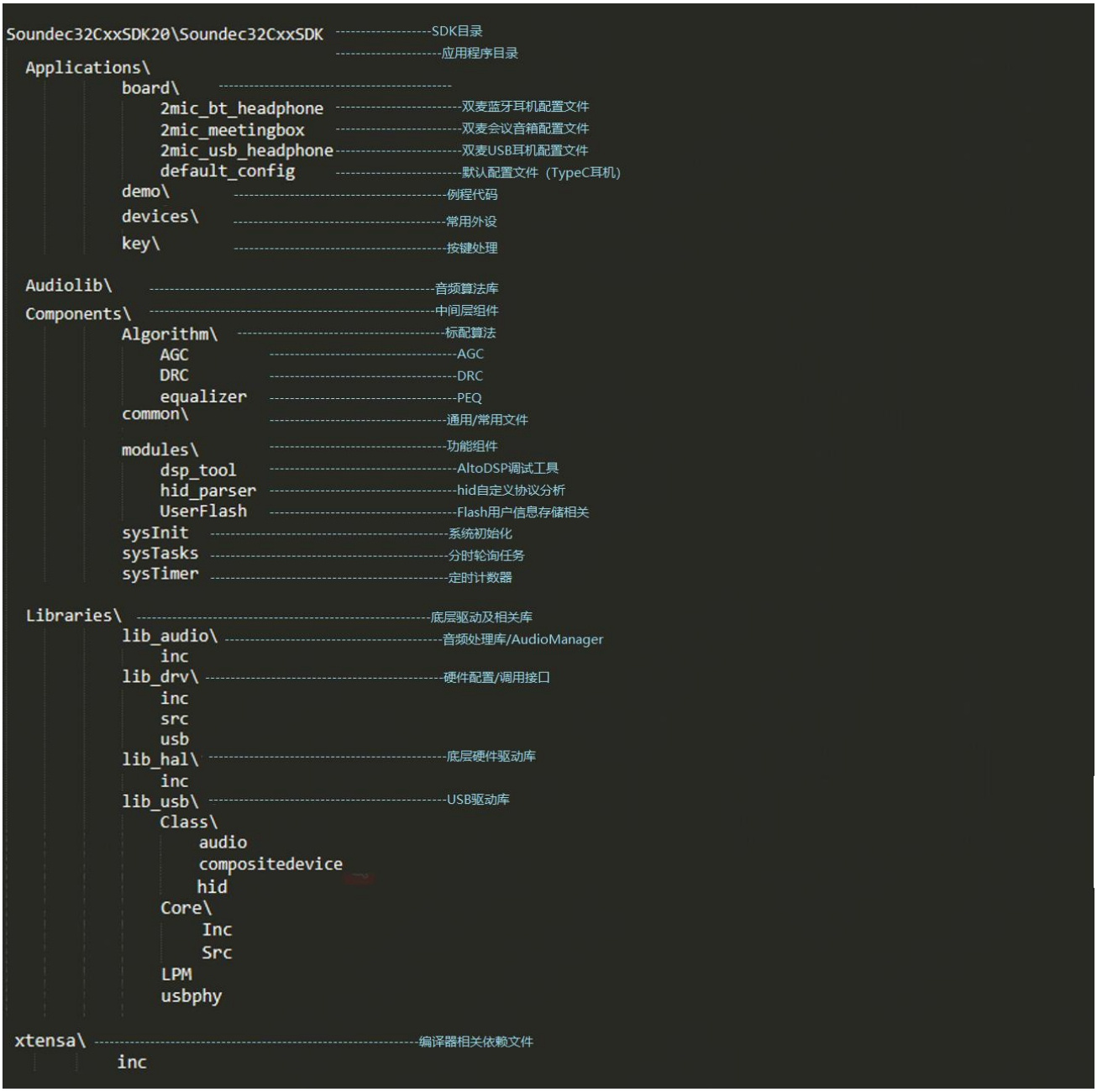


1.3 主要文件目录简介

Soundec32CxxSDK 目录下主要包含 Application、Audiolib、Components、Libraries、xtensa 5 个目录，用户主要通过修改和扩展 Application 目录下的文件来实现项目的二次开发。

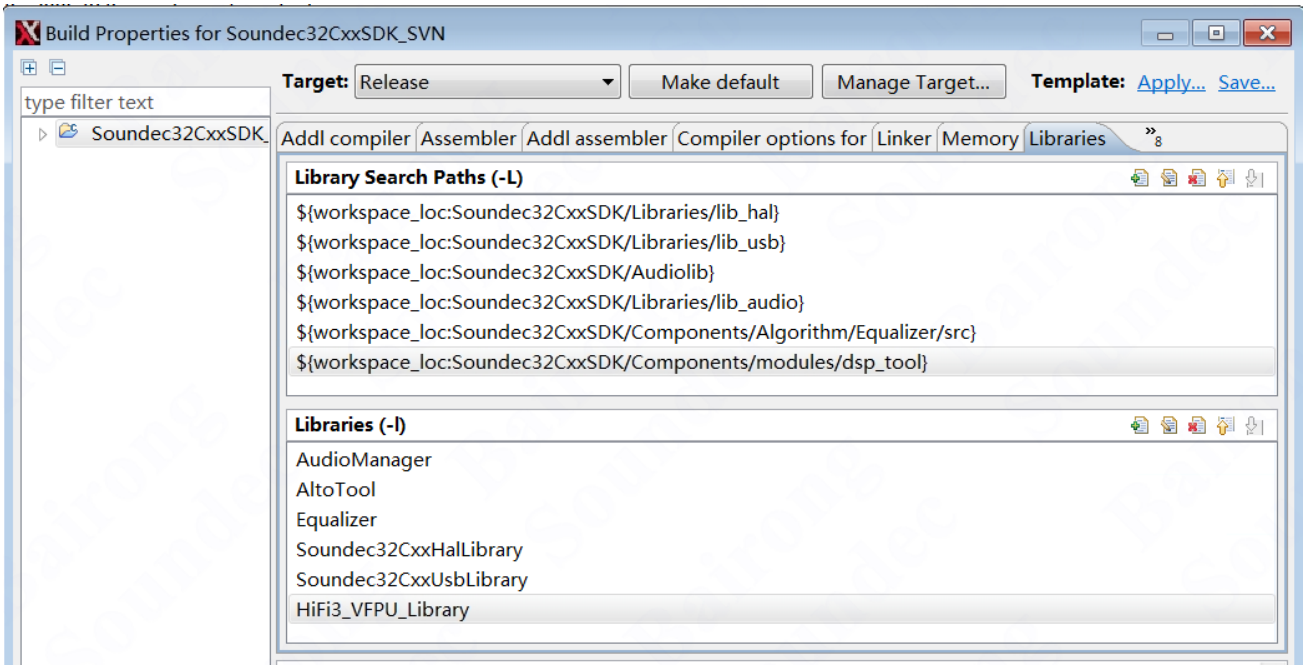
Application 目录中，用户主要需要修改的内容如下：

- Board 目录用于扩展用户应用
- Demo 目录下提供 SNC86xx 支持的非音频外设的使用例程
- Key 目录下提供 ADC 和 HID 按键的使用例程



1.3.1 主要库文件的添加

随 SDK 发行的主要库文件，分别介绍如下
lib_hal 和 lib_usb 驱动库库为必须添加项；



1.3.2.1 必要的库文件

硬件驱动库 libSoundec32CxxHalLibrary.a

路径：…/Soundec32CxxSDK/Libraries/lib_hal/libSoundec32CxxHalLibrary.a

USB 驱动库 libSoundec32CxxHalLibrary.a

路径：…/Soundec32CxxSDK/Libraries/lib_usb/libSoundec32CxxUsbLibrary.a

1.3.2.2 算法库文件

如果应用需要支持算法处理，则需要包含 HiFi3 浮点库：

HiFi3 浮点库 libHiFi3_VFPU_Library.a

路径：…/Soundec32CxxSDK/Audiolib/libHiFi3_VFPU_Library.a

如果需要支持 EQ，则需要包含 EQ 算法库：

EQ 库 libEqualizer.a

路径：…/Soundec32CxxSDK/Components/Algorithm/Equalizer/src/libEqualizer.a

2 扩展一个应用

2.1 新增项目配置

根据不同的硬件设计及资源，配置板级资源：boardConfig.h，在其中定义并使能项目配置

#define D_CONFIG_XXXX 1

```
 * @file name    : boardConfig.h

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __BOARDCONFIG_H__
#define __BOARDCONFIG_H__

#define D_CONFIG_2MIC_MEETING_BOX               0
#define D_CONFIG_2MIC_USB_HEADPHONE             0
#define D_CONFIG_2MIC_BT_HEADPHONE              0

#if D_CONFIG_2MIC_MEETING_BOX
#define PROJECT_NAME                            "2mic_meeting_box"
#include "user_config_2mic_meetingbox.h"
#elif D_CONFIG_2MIC_USB_HEADPHONE
#define PROJECT_NAME                            "2mic_usb_headphone"
#include "user_config_2mic_usb_headphone.h"
#elif D_CONFIG_2MIC_BT_HEADPHONE
#define PROJECT_NAME                            "2mic_bt_headphone"
#include "user_config_2mic_bt_headphone.h"
#else
#define D_CONFIG_DEFAULT                        1
#define PROJECT_NAME                            "Soundec_default"
#include "user_config_default.h"
#endif


#endif  /* __BOARDCONFIG_H__ */
```

2.2 新增配置文件和音频描述文件

为了支持客制化应用的快速开发，参考 default_config 目录，创建配置文件和音频描述文件：

- 配置文件：user_config_xxx.h
- 音频描述文件：audio_session_desc_xxx.c

2.3 修改配置文件

以默认配置文件 user_config_default.h 为例，根据以下模块说明配置相关功能。详细功能开关说明可以参考对应的 ApplicationNote 文件。

- 修改项目版本号

```
/*****************************************************
 *  Version config
 *****************************************************/
#define CONFIG_PRODUCT_VERSION              0x200B
```

- 选择芯片型号

```
/*****************************************************
 *  Chip config
 *****************************************************/
#define CONFIG_CHIP_TYPE                    CHIP_SNC8600
```

- 配置晶振及系统时钟

```
/*****************************************************
 *   Module:      PLL
 *****************************************************/
#define SYSTEM_CRYSTAL                  SYSTEM_CRYSTAL_24MHZ
// [PLL]
#define PLL_OUT_VALUE                   PLLOUT_190MHZ
```

- 配置 Uart

```
/*****************************************************
 *   Module:      UART
 *****************************************************/
#define UART_ENABLE                         1
#if(UART_ENABLE == 1)
//user code
#define UART_BUARD_RATE                     BAUDRATE_2M/ *BAUDRATE_9600*/
#define UART_INTE_ENABLE                    0
#define UART_RECEIVE                        0
#define UART_QUEUE_SIZE                     1024
#endif
```

- 配置 Codec

```
/*****************************************************
 *   Module:      Codec
 *****************************************************/
#define CODEC_ENABLE                    1
#if (CODEC_ENABLE)
#define CODEC_ADC_ENABLE                0
#define CODEC_ADC12_ENABLE              1
#define CODEC_ADC34_ENABLE              0
#define CODEC_ADC56_ENABLE              0
#define CODEC_ADC78_ENABLE              0
#define CODEC_ADC9A_ENABLE              0

#define CODEC_DAC_ENABLE                1

#define CODEC_MASTER                    1       /*SLAVE MODE:ADC12 must be initialized, or DAC can not work normally.*/
#define CODEC_ADC_DATA_MODE             CODEC_ADC_DATA_MODE_24BIT
#define CODEC_SLAVE_FRAC_SRC            CODEC_SLAVE_FRAC_SRC_SNC
#define CODEC_SLAVE_FRAC_SHARE          CODEC_SLAVE_FRAC_SHARE_DEDICATED
#define CODEC_FIFO_AE_LEVEL             8       /*Almost empty level*/
#if(DMA_ENABLE && AUDIO_MANAGER)
#define CODEC_FIFO_AF_LEVEL             1//8        /*Almost full level*/
#else
#define CODEC_FIFO_AF_LEVEL             8       /*Almost full level*/
#endif  //DMA_ENABLE && AUDIO_MANAGER
#define ADC_MICBIAS                     1
//#define ADC_DMA_ENABLE
//#define CODEC_ENABLE_MIC_AGC
//#define CODEC_ENABLE_SPK_DRC
#define DAC_AIAS                        1
#define ADCX_AIAS                       0
#define ADC12_AIAS                      0
#define ADC3456_AIAS                    0
#define ADC789a_AIAS                    0

#define CONFIG_CODEC_FREQUENCY          SAMPLING_RATE_48000
#endif
```

- 配置 USB

```
/*****************************************************
 *  Module:     USB
 *****************************************************/
#define USB_ENABLE                          1
#if USB_ENABLE

#define SUPPORT_UAC20

#define USB_SPEED_CFG                       USB_SPEED_FULL

#define SUPPPORT_HID_KEYBOARD
#define SUPPORT_USB_SPEAKER                 1
#define SUPPORT_USB_MIC                     1

#if ((SUPPORT_USB_SPEAKER)&&(SUPPORT_USB_MIC)&&(USB_SPEED_CFG == USB_SPEED_HIGH))      //MIXER
#define SUPPORT_SPEAKER_SIDETONE            0
#else
#define SUPPORT_SPEAKER_SIDETONE            0    //must set to 0.
#if SUPPORT_SPEAKER_SIDETONE
######"error."
#endif
#endif

//#define ALTO_USB_LPM_ENABLE
#define CONGIFG_USB_DEV_VOLUME_FIT_USBH

#ifdef ALTO_USB_LPM_ENABLE
#define HANDLE_AO_ADC_KEY_WAKEUP_FROM_USB_DEV_SUSPEND
#define HANDLE_USB_DEV_SUSPEND_MODE2
#endif

//#define USB_DEV_KEYSIGHT_TEST_MODE
#define SUPPORT_SYNC_HOST_VOL
#define SUPPORT_UAC20
#define SUPPORT_UAC20_MIC_MULTIPLE_CHANNEL
//#define CONGIFG_USB_DEV_VOLUME_FIT_USBH

#define CONFIG_USB_SPK_FREQUENCY            SAMPLING_RATE_48000
#define CONFIG_USB_MIC_FREQUENCY            SAMPLING_RATE_48000

#endif
```

- 配置 I2S

```
/*****************************************************
 *  Module:     I2S
 *****************************************************/
#define I2S_ENABLE                          1
#if (I2S_ENABLE == 1)
#define I2S1_ENABLE                         0
#define I2S2_ENABLE                         0
#define I2S3_ENABLE                         0
#define I2S_TX_FIFO_LEVEL                   8  /*1-16*/
#define I2S_RX_FIFO_LEVEL                   8  /*1-16*/
#if(DMA_ENABLE)
#define I2S_DMA_ENABLE
#endif
#endif
```

- 配置 DMA

```
/*****************************************************
 *  Module:     DMA
 *****************************************************/
#define DMA_ENABLE                          0
#define DMAC_CHX_NULL                       6
#if(DMA_ENABLE)
#if(AUDIO_MANAGER)
#define DMA_CHX_MEM_TO_MEM                  0
#define DMA_CHX_MEM_TO_I2S1                 DMAC_CHANNEL_1
#define DMA_CHX_ADC_TO_MEM                  DMAC_CHANNEL_5
#else
#define DMA_TEST_MODE_ENABLE                0
#if(DMA_TEST_MODE_ENABLE)
#define DMA_TEST_I2S_USB_ENABLE             0
#define DMA_TEST_CODEC_ADC_ENABLE           1
#endif  //DMA_TEST_MODE_ENABLE
#endif  //AUDIO_MANAGER
#endif  //DMA_ENABLE
```

■ 配置 Flash

```
/*****************************************************
 *  Module:     Flash
 *****************************************************/
#define FLASH_ENABLE                    0
#if(FLASH_ENABLE)
#define FLASH_DEBUG_ENABLE              1
#define FLASH_WRITE_PROTECT_ENABLE      0//1
#define FLASH_SFC_CACHE_ENABLE          0
#define FLASH_SAVE_BANK_ENABLE          1
#if(FLASH_SAVE_BANK_ENABLE)
#define FLASH_SAVE_BANK_DEBUG_ENABLE    1
#endif
#define FLASH_GET_UID_ENABLE            1
#if(FLASH_GET_UID_ENABLE)
#define FLASH_UID_AS_USB_SN_ENABLE      1
#endif
#endif
```

■ 配置 ADC

```
/*****************************************************
 *  Module:     AUX ADC
 *****************************************************/
#define AUX_ADC_ENABLE                  0
#if(AUX_ADC_ENABLE)
#define AUX_ADC_INTR_ENABLE             0
#define AUX_ADC_USE_CH0                 1   //if 0, use ch1
#define APP_ADC_KEY_ENABLE              1
#endif  //AUX_ADC_ENABLE
```

■ 配置 PWM

```
/*****************************************************
 *  Module:     PWM
 *****************************************************/
#define PWMX_ENABLE                     0
#if PWMX_ENABLE == 1
//use code
#define PWM_NORMAL_ENABLE               1
#define PWM_GPIO_MODE_ENABLE            1
#define PWM_TIMER1_MODE_ENABLE          1 //count a
#define PWM_TIMER2_MODE_ENABLE          1 //cycle count

#define PWM_INT_ENABLE                  1
#endif
```

■ 配置 I2C

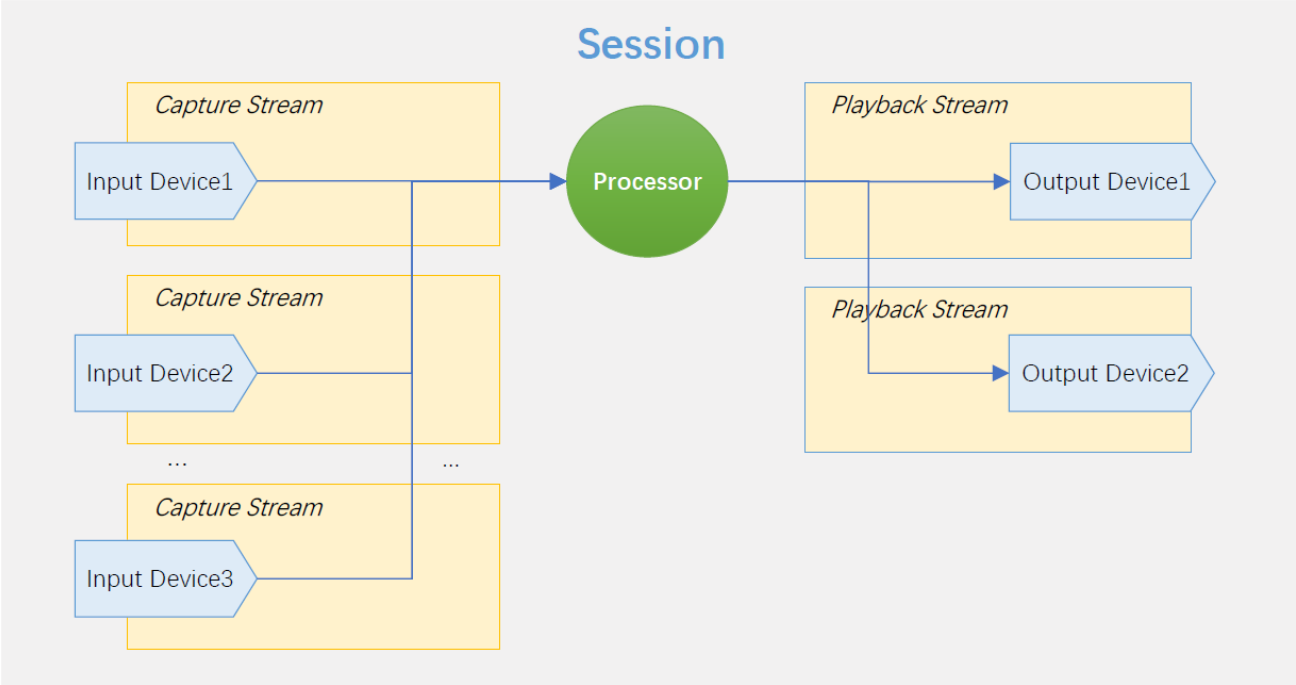
```
/*****************************************************
 *  Module:     I2C
 *****************************************************/
#define I2C_ENABLE                      0
#if I2C_ENABLE == 1
#define I2C1_ENABLE                     1

#if(I2C1_ENABLE == 1)
//user code
#define I2C1_INTE_ENABLE
#define I2C1_MODE                       I2C_MASTER
#endif

#define I2C2_ENABLE                     1
#if(I2C2_ENABLE == 1)
//user code
#define I2C2_INTE_ENABLE
#define I2C2_MODE                       I2C_SLAVE
#endif
#endif
```

2.4 修改音频通路描述文件

根据音频通路模型，描述每一路音频配置。每个通路均支持多路音频输入和多路音频输出，用户根据应用需求分别在 capture 和 playback 中描述输入和输出流的信息。通路简介见下图说明



2.4.1 描述音频信号通路

以双麦会议音箱应用为例，说明音频描述文件 audio_session_desc_2mic_meetingbox.c 的配置。此配置支持两路音频通路：AUDIO_SESSION_0 和 AUDIO_SESSION_1。

- AUDIO_SESSION_0 支持 2 路输入（capture），1 路输出（playback），输入信号经过（session0Proc）处理后输出；
- AUDIO_SESSION_1 支持 1 路输入（capture），1 路输出（playback），输入信号不需要处理，直接输出：

```
auSessionDesc_t auSessionDesc[AUDIO_SESSION_MAX] = {
    {
        .id = AUDIO_SESSION_0,
        .sessionPolicy = &session0Policy,
        .captureNum = 2,
        .capture = session0StreamIn,

        .procIf = &session0Proc,

        .playbackNum = 1,
        .playback = session0StreamOut,

    },
    {
        .id = AUDIO_SESSION_1,
        .captureNum = 1,
        .capture = session1StreamIn,

        .procIf = NULL,

        .playbackNum = 1,
        .playback = session1StreamOut,

    },
};
```

2.4.2 配置音频信号

以双麦会议音箱应用为例，举例说明如何配置音频流的输入、处理和输出信息。

2.4.2.1 音频流的输入配置——Capture

AudioManager 使用 Capture 相关定义描述输入源信息，其中：
mems12_in 作为音箱参考信号的输入源，mems34_in 作为两路数字麦克风的输入源;

```
auCaptureIf_t *session0StreamIn[]=
{
    &mems12_in,
    &mems34_in,
};
```

通过 auCaptureIf_t 配置接口，分别描述输入设备的硬件信息和音频格式，以 mems12_in 为例：

```
auCaptureIf_t mems12_in     = {
    .capture = {
        .devId              = AUDIO_DEVICE_MEMS12_IN,
        .type               = STREAM_VOICE,
    },
    .format = {
        .channlesPerFrame   = 2,
        .frameLength        = 20,
        .pcmFormat          = PCM_PLANAR_FORMAT,
        .bitSlot            = BIT_SLOT_32,
        .sampleRateHz       = SAMPLING_RATE_48000,
        .sampleRateHzMax    = SAMPLING_RATE_48000,
    },
};
```

1） Capture 输入源配置
   a) devId 选择硬件设备为 ADC12（AUDIO_DEVICE_ MEMS12_IN）
   b) 音频流类型为 Voice
2） Format 音频格式
   a) channelsPerFrame = 2 双声道
   b) frameLength = 20，注意此处 frameLength 以时间 ms（毫秒）为单位，用于表示 Mems12 的数据缓存区大小
   c) pcmFormat = PCM_PACKED_FORMAT，描述音频流的数据格式，支持两种存放格式：
      ◆ 声道交错存储：PCM_PACKED_FORMAT，如 LRLR…LR
      ◆ 声道独立存储：PLANAR_FORMAT，如 LLL…RRR
   d) bitSlot：默认数据位宽
   e) sampleRateHz：默认采样率，单位为 Hz
   f) sampleRateHzMax：最大采样率，单位为 Hz

[注意]：如果输入信号需要算法处理，音频格式根据算法需求配置

2.4.2.2 音频流的处理配置——Proc

AudioManager 使用 Proc 相关定义描述音频流的处理信息：
mems12_in 和 mems34_in 信号送入 session0Proc 中描述的 onFifoReady 回调函数进行处理。
通过 auProcIf_t 接口，描述当前 session 通路上的算法信息，其中：
proEvent 用于描述算法对输入端的信号需求；outputFormat 用于描述算法输出格式信息。

```
auProcIf_t session0Proc = {
    .procEvent = {
        .frameThreshold = 5,
        .onFifoReady = meetingBox_onFifoReady,
        .channelCopy = FALSE,
        .irqNum = SW_IRQn,
    },
    .outputFormat = {
        .sampleRateHz       = SAMPLING_RATE_48000,
        .channlesPerFrame   = 1,
        .wordSize           = BIT_SLOT_32,//fifo push pop
        .bitSlot            = BIT_SLOT_32,
        .pcmFormat          = PCM_PACKED_FORMAT,
        .frameLength        = 20,
    },
};
```

2.4.2.3 音频流的输出配置——Playback

AudioManager 使用 Playback 相关定义描述音频流的输出信息，以下描述表示 Session0 的信号通过 USB 录音通道输出。

```
auPlaybackIf_t *session0StreamOut[] = {
    &usb_record,
};
```

通过 auPlaybackIf_t 配置接口，描述输出设备的硬件信息、音频格式，以 usb_record 为例，其中关于 Playback 的设备信息和音频格式描述参考 Capture，

```
auPlaybackIf_t usb_record    = {
    .playback = {
        .devId = AUDIO_DEVICE_USB,
        .type = STREAM_VOICE,   /*reserved*/
    },
    .format = {
        .channlesPerFrame   = 2,
        .frameLength        = 10,   /*not used*/
        .pcmFormat          = PCM_PACKED_FORMAT,
        .bitSlot            = BIT_SLOT_24,
        .sampleRateHz       = SAMPLING_RATE_48000,
        .sampleRateHzMax    = SAMPLING_RATE_192000, /*not used*/
    },
    .channelCopy = TRUE,
};
```

另多出一项配置 channelCopy：用于支持声道复制功能，配置此项为 true 表示将单声道音频源用双声道输出

2.4.2.4 配置通路的通用属性

每个通路通过 auPolicyPatam_t 结构体来定义和存储通路上的音频设置和处理控制，操作对整个音频通路(Session)有效。部分成员定义未使用功能，用于以后扩展。

```
auPolicyParam_t session0Policy ={
    .interruptUnmixedSession = FALSE,   /*reserved*/
    .syncSamplerate = TRUE,
    .procBypass = FALSE,

    .bitSlot = BIT_SLOT_24,     /*default*/
    .sampleRate = 48000,        /*default*/
    .maxSampleRate = 192000,    /*reserved*/
    .maxLatencyInFrames = 0,    /*reserved*/
};
```

3 其他功能说明

3.1 按键扩展

参考应用手册《SNC86xx_ADC_KEY_ApplicationNote_V0.1》

3.2 Task 扩展

本 SDK 使用系统定时器 Timer2 作为轮询时钟源，支持最小 1ms 的轮询时间片，用于扩展简单的轮询任务。

Task 任务扩展参考注册接口 task_register()：

```
void task_register(uint8_t index, int itvl_ms, task_hook_t task);
```

1） 根据注册的 task 任务总数，首先扩展 TASK_NUM

```
/* Private macro -------------------------------------
#define TASK_NUM    3
```

2） taskn_init 初始化完成之后，根据需要注册对应的 task，参数说明如下：

a) index：每注册一个 task，分配 index 序号，取值范围 0~index，index<TASK_NUM
b) itvl_ms：被注册大 task 的轮询间隙，单位 ms；注册之后，系统将每间隔 itvl_ms 来调用 task
c) task：注册的 task 的回调函数，被轮询调用的实体

4 附录

4.1 名词解释

AudioManager 是 **Soundec32CxxSDK** 的音频架构，其中：
Capture(Stream) 表示音频流的输入流
Playback(Stream) 表示音频流的输出流
Session 表示一个完整的音频流的输入输出通路