ANC接口

cpu\br36\audio\audio_anc.h

ANC模式定义

```
#define ANC_OFF_BIT          BIT(1)  /* 降噪关闭使能 */
#define ANC_ON_BIT           BIT(2)  /* 降噪模式使能 */
#define ANC_TRANS_BIT        BIT(3)  /* 通透模式使能 */
```

模式枚举

```
static const char *anc_mode_str[] = {
    "NULL",          /* 无效/非法 */
    "ANC_OFF",       /* 关闭模式 */
    "ANC_ON",        /* 降噪模式 */
    "Transparency", /* 通透模式 */
    "ANC_BYPASS",    /* BYPASS模式 */
    "ANC_TRAIN",     /* 训练模式 */
    "ANC_TRANS_TRAIN",  /* 通透训练模式 */
};
```

主要接口函数

模式控制

- `void anc_mode_switch(u8 mode, u8 tone_play);` - 切换到指定模式，并可选择是否播放提示音
- `void anc_mode_next(void);` - 切换到下一个模式
- `u8 anc_mode_get(void);` - 获取当前ANC模式
- `u8 anc_mode_sel_get(void);` - 获取模式选择

初始化与电源管理

- `void anc_init(void);` - ANC初始化
- `void anc_poweron(void);` - 上电
- `void anc_poweroff(void);` - 下电
- `void anc_suspend(void);` - 挂起
- `void anc_resume(void);` - 恢复

训练模式

- `void anc_train_open(u8 mode, u8 debug_sel);` - 打开训练模式
- `void anc_train_close(void);` - 关闭训练模式
- `int anc_train_open_query(void);` - 查询是否处于训练状态

增益控制

- `u8 anc_dac_gain_get(u8 ch);` - 获取DAC增益
- `u8 anc_mic_gain_get(void);` - 获取MIC增益
- `void anc_dynamic_micgain_start(u8 audio_mic_gain);` - 开始动态MIC增益
- `void anc_dynamic_micgain_stop(void);` - 停止动态MIC增益

回调与配置

- `void anc_api_set_callback(void (*callback)(u8, u8), void (*pow_callback)(anc_ack_msg_t *, u8));` - 设置回调函数
- `void anc_api_set_fade_en(u8 en);` - 设置淡入淡出使能
- `void anc_mode_enable_set(u8 mode_enable);` - 设置支持的模式

配置选项

```
#define ANC_COEFF_SAVE_ENABLE    1   /* ANC滤波器表保存使能 */
#define ANC_INFO_SAVE_ENABLE     0   /* ANC信息记忆 */
#define ANC_TONE_PREEMPTION      0   /* ANC提示音打断播放(1)还是叠加播放(0) */
#define ANC_FADE_EN              1   /* ANC淡入淡出使能 */
#define ANC_MODE_SYSVDD_EN       1   /* ANC模式提高SYSVDD */
#define ANC_TONE_END_MODE_SW     1   /* ANC提示音结束进行模式切换 */
#define ANC_MODE_FADE_LVL        1   /* 降噪模式淡入步进 */
#define ANC_HEARAID_EN           0   /* ANC辅听器使能 */
#define ANC_HEARAID_HOWLING_DET  1   /* ANC辅听器啸叫抑制使能 */
#define ANC_AHS_EN               1   /* 回声抑制使能:防啸叫 */
#define ANC_CMP_EN               1   /* 音乐补偿使能 */
#define ANC_DRC_EN              0    /* ANC DRC使能 */
```

其他功能

- 支持TWS同步
- 支持在线调试
- 支持音乐动态增益调节
- 支持辅听器模式
- 支持参数保存和恢复

这个头文件定义了ANC功能的核心接口和配置选项，包括模式控制、增益调节、训练模式等功能。您可以使用这些接口来实现ANC的开启、关闭、模式切换等操作。

常用接口

`void anc_mode_next(void);` - 切换到下一个模式

```
/**
 * @brief 切换到下一个ANC模式
 *
 * 该函数用于在已启用的ANC模式之间循环切换，切换顺序由anc_mode_switch_tab数组定义。
 * 切换时会自动跳过未启用的模式，并播放提示音。
 *
```

```
 * 工作流程：
 * 1. 检查ANC句柄是否存在
 * 2. 检查是否处于训练模式，是则直接返回
 * 3. 检查经典蓝牙是否已连接，未连接则直接返回
 * 4. 查找当前模式在模式切换表中的位置
 * 5. 计算下一个模式，并确保是已启用的模式
 * 6. 执行模式切换并保存设置
 */
void anc_mode_next(void)
{
    if (anc_hdl) {  // 检查ANC句柄是否存在
        // 如果处于训练模式，则直接返回，不允许切换模式
        if (anc_train_open_query()) {
            return;
        }

        // 只有在经典蓝牙已连接的情况下才允许切换模式
        if (get_tws_phone_connect_state() == FALSE)
        {
            return;
        }

        u8 next_mode = 0;
        ucAnc_change_mode = 1;  // 设置模式切换标志

        // 关中断，防止在模式切换过程中被中断
        local_irq_disable();

        // 确保淡入淡出功能开启
        anc_hdl->param.anc_fade_en = ANC_FADE_EN;

        // 查找当前模式在模式切换表中的位置
        for (u8 i = 0; i < ANC_MODE_NUM; i++) {
            if (anc_mode_switch_tab[i] == anc_hdl->param.mode) {
                // 计算下一个模式索引
                next_mode = i + 1;

                // 如果超出模式数量，则回到第一个模式
                if (next_mode >= ANC_MODE_NUM) {
                    next_mode = 0;
                }

                // 检查下一个模式是否已启用，如果未启用则继续查找下一个
```

```
                if ((anc_hdl->mode_enable &
BIT(anc_mode_switch_tab[next_mode])) == 0) {
                    user_anc_log("anc_mode_filt,next:%d,en:%d", next_mode,
anc_hdl->mode_enable);
                    next_mode++;
                    if (next_mode >= ANC_MODE_NUM) {
                        next_mode = 0;
                    }
                }
                break;
            }
        }

        // 恢复中断
        local_irq_enable();

        // 记录模式切换日志
        user_anc_log("anc_next_mode:%d old:%d,new:%d", next_mode, anc_hdl-
>param.mode, anc_mode_switch_tab[next_mode]);

        // 获取新的模式值
        u8 new_mode = anc_mode_switch_tab[next_mode];
        user_anc_log("new_mode:%s", anc_mode_str[new_mode]);

        // 执行模式切换，并播放提示音
        anc_mode_switch(anc_mode_switch_tab[next_mode], 1);

        // 保存ANC模式设置（减1是因为保存的索引从0开始）
        save_anc_voice_ex(new_mode-1);
    }
}
```

模式切换说明

根据代码分析，`anc_mode_next`函数会按照`anc_mode_switch_tab`数组中定义的顺序切换模式。虽然完整的`anc_mode_switch_tab`定义在代码的其他部分，但从上下文和`audio_anc.h`中的定义可以推断，它可能包含以下几种模式（按顺序）：

1. **关闭模式 (ANC_OFF)** - 关闭主动降噪功能
2. **降噪模式 (ANC_ON)** - 开启主动降噪
3. **通透模式 (Transparency)** - 允许外部声音进入

4. **BYPASS模式 (ANC_BYPASS)** - 直通模式
5. **训练模式 (ANC_TRAIN)** - 用于ANC参数训练
6. **通透训练模式 (ANC_TRANS_TRAIN)** - 通透模式的训练

切换逻辑

1. 函数会跳过未启用的模式（通过`anc_hdl->mode_enable`位掩码判断）
2. 如果当前模式是最后一个模式，则循环回到第一个模式
3. 切换时会自动播放提示音（`anc_mode_switch`的第二个参数为1表示播放提示音）
4. 切换完成后会保存当前模式设置

注意事项

1. 模式切换需要满足以下条件：
    - ANC句柄已初始化
    - 不处于训练模式
    - 已连接经典蓝牙
2. 模式切换是原子操作，会暂时关闭中断以确保数据一致性
3. 切换过程中会记录详细的日志，方便调试

ANC的记忆功能

ANC模式保存

- 在`anc_info_save()`函数中，系统会将当前ANC模式保存到配置中
- 使用`syscfg_write(CFG_ANC_INFO, &anc_info, sizeof(anc_info))`将ANC信息写入持久化存储
- 保存的信息包括：
    - `mode`：当前ANC模式
    - `mode_enable`：已启用的模式

- `inear_tws_mode`：入耳检测TWS模式（如果启用）

ANC模式恢复

- 在`anc_poweron()`函数中，系统会从持久化存储中读取之前保存的ANC模式
- 使用`syscfg_read(CFG_ANC_INFO, &anc_info, sizeof(anc_info))`读取保存的信息
- 如果读取成功且当前ANC状态为`ANC_STA_INIT`（初始化状态）且之前保存的模式不是`ANC_OFF`，则自动切换到之前保存的模式
- 通过调用`anc_mode_switch(anc_info.mode, 0)`进行模式切换，第二个参数0表示不播放提示音

调用时机

- `anc_poweron()`函数在蓝牙连接状态变化时被调用（在`bt_connction_status_event_handler`函数中）
- 当蓝牙初始化完成或连接状态变化时，系统会尝试恢复之前保存的ANC模式

关键宏定义

```
#define ANC_INFO_SAVE_ENABLE 0  /* ANC信息记忆:保存上一次关机时所处的降噪模式等 */
```

注意：这个宏默认被设置为0，表示默认情况下ANC记忆功能是禁用的。要启用ANC记忆功能，需要将此宏设置为1。

数据结构

```
typedef struct {
    u8 mode;        /* 当前模式 */
    u8 mode_enable; /* 使能的模式 */
#if INEAR_ANC_UI
    u8 inear_tws_mode;
#endif
} anc_info_t;
```

总结

ANC模式记忆功能默认是禁用的，需要将`ANC_INFO_SAVE_ENABLE`宏设置为1来启用

1. 启用后，系统会在模式切换时自动保存当前模式到持久化存储
2. 在蓝牙初始化或连接状态变化时，系统会自动恢复之前保存的ANC模式
3. 恢复模式时不会播放提示音，避免干扰用户体验

如果您需要启用ANC记忆功能，需要修改`audio_anc.h`文件中的`ANC_INFO_SAVE_ENABLE`宏定义，将其值从0改为1。

流程图说明

mermaid源码：

```
graph TD
    %% ==================== 流程开始 ====================
    START([开始]) --> A([系统上电])

    %% ==================== 系统启动流程 ====================
    A --> B[app_main系统初始化]
    B --> C[蓝牙初始化完成]
    C --> D[BT_STATUS_INIT_OK事件]
    D --> E[earphone.c中调用anc_poweron]
    E --> F{TCFG_AUDIO_ANC_ENABLE?}
    F -->|是| G[进入anc_poweron函数]
    F -->|否| END1([ANC功能未启用<br/>流程结束])

    %% ==================== ANC初始化流程 ====================
    G --> H{anc_hdl有效?}
    H -->|否| END2([ANC初始化失败<br/>流程结束])

    %% ==================== 记忆功能开启路径 ====================
    H -->|是| I{ANC_INFO_SAVE_ENABLE?}
    I -->|是| J[从NVRAM读取保存的ANC配置]
    J --> K{读取成功?}
    K -->|是| L[检查当前状态]
    K -->|否| M[使用默认配置<br/>设置模式为ANC_OFF]
    L --> N{状态==INIT且保存的模式!=OFF?}
```

```
    N -->|是| O[切换到保存的模式,不播放提示音]
    N -->|否| P[保持当前状态]

    %% ==================== 记忆功能关闭路径 ====================
    I -->|否| M

    %% ==================== 运行中模式切换 ====================
    O --> Q[ANC正常运行]
    P --> Q
    M --> Q

    Q --> R{用户切换模式?}
    R -->|是| S[调用anc_mode_switch]
    S --> T[播放提示音]
    T --> U[执行模式切换]
    U --> V{ANC_INFO_SAVE_ENABLE?}
    V -->|是| W[保存新配置到NVRAM]
    V -->|否| X[不保存配置]
    W --> Q
    X --> Q

    %% ==================== 系统关机流程 ====================
    R -->|否| Y{系统准备关机?}
    Y -->|否| Q
    Y -->|是| Z[调用anc_poweroff]
    Z --> AA{ANC_INFO_SAVE_ENABLE?}
    AA -->|是| BB[保存当前状态到NVRAM]
    AA -->|否| CC[不保存状态]
    BB --> DD[执行ANC淡出关闭]
    CC --> DD
    DD --> END3([系统关机完成])

    %% ==================== 样式定义 ====================
    classDef startEnd fill:#e1f5fe,stroke:#01579b,stroke-width:3px,color:#000
    classDef condition fill:#fff3e0,stroke:#e65100,stroke-width:2px,color:#000
    classDef process fill:#f3e5f5,stroke:#4a148c,stroke-width:2px,color:#000
    classDef save fill:#e8f5e8,stroke:#2e7d32,stroke-width:2px,color:#000
    classDef nosave fill:#ffebee,stroke:#c62828,stroke-width:2px,color:#000
    classDef running fill:#e3f2fd,stroke:#1565c0,stroke-width:2px,color:#000
    classDef error fill:#ffcdd2,stroke:#d32f2f,stroke-width:2px,color:#000

    %% ==================== 应用样式 ====================
    class START,END1,END2,END3 startEnd
```

```
    class F,H,I,K,N,R,Y,V,AA condition
    class A,B,C,D,E,G,J,L,O,P,M,S,T,U,Z,BB,CC,DD process
    class W,BB save
    class X,CC nosave
    class Q running
```

anc_poweron

`apps\earphone\earphone.c`的`bt_connction_status_event_handler`当状态机执行了初始化蓝牙协议栈之后，初始化完毕就会进入到`BT_STATUS_INIT_OK`分支，通过条件编译`TCFG_AUDIO_ANC_ENABLE`是否使能ANC决定是否调用`anc_poweron();`

```
/* 系统上电时，根据NVRAM中保存的配置决定是否进入上次的ANC模式 */
void anc_poweron(void)
{
    if (anc_hdl) {  // 检查ANC句柄是否有效
#if ANC_INFO_SAVE_ENABLE  /* 记忆功能使能宏
                          * 1: 使能ANC模式记忆功能，系统会保存最后一次的ANC模式到
NVRAM
                          * 0: 禁用记忆功能，每次开机都使用默认的ANC_OFF模式
                          */
        anc_info_t anc_info;  // 定义ANC信息结构体
        // 从NVRAM中读取保存的ANC配置信息
        int ret = syscfg_read(CFG_ANC_INFO, &anc_info, sizeof(anc_info));
        if (ret == sizeof(anc_info)) {  // 读取成功
            // 打印当前ANC状态和读取到的模式信息
            user_anc_log("read anc_info succ,state:%s,mode:%s",
                         anc_state_str[anc_hdl->state],
                         anc_mode_str[anc_info.mode]);

#if INEAR_ANC_UI  /* 入耳检测UI使能宏
                  * 1: 支持入耳检测UI功能
                  * 0: 不支持入耳检测UI功能
                  * 当使能时，需要同步TWS耳机的ANC模式
                  */
            inear_tws_ancmode = anc_info.inear_tws_mode;  // 同步TWS耳机的ANC模
式
#endif/*INEAR_ANC_UI*/

            // 检查当前状态是否为初始化状态，并且保存的模式不是关闭状态
```

```
            if ((anc_hdl->state == ANC_STA_INIT) && (anc_info.mode !=
ANC_OFF)) {
                anc_mode_switch(anc_info.mode, 0);  // 切换到保存的模式，不播放提
示音
            }
        } else {
            user_anc_log("read anc_info err");  // 读取失败，记录错误日志
        }
#endif/*ANC_INFO_SAVE_ENABLE*/
    }
}
```

关键宏说明

1. **ANC_INFO_SAVE_ENABLE**
    - **作用**：控制是否启用ANC模式记忆功能
    - 当设置为1时：
        - 系统会保存最后一次的ANC模式到NVRAM
        - 下次开机时会自动恢复到最后一次使用的模式
    - 当设置为0时：
        - 不会保存ANC模式信息
        - 每次开机都使用默认的ANC_OFF模式
2. **INEAR_ANC_UI**
    - **作用**：控制是否启用入耳检测UI功能
    - 当设置为1时：
        - 支持入耳检测相关的UI功能
        - 会同步TWS耳机的ANC模式
    - 当设置为0时：
        - 不处理入耳检测相关的UI逻辑
        - 不同步TWS耳机的ANC模式
3. **CFG_ANC_INFO**
    - **作用**：NVRAM中保存ANC信息的配置项标识
    - 用于在NVRAM中唯一标识ANC配置数据
4. **ANC_STA_INIT**
    - **作用**：表示ANC模块的初始化状态

- 只有在初始化状态下才会尝试恢复之前的模式

5. **ANC_OFF**
    - **作用**：表示ANC关闭状态
    - 如果保存的模式是关闭状态，则不需要进行模式切换

这个函数的执行流程是：

1. 检查ANC句柄是否有效
2. 如果启用了记忆功能，则从NVRAM读取保存的配置
3. 读取成功且当前处于初始化状态时，恢复到最后一次使用的模式
4. 如果启用了入耳检测UI，则同步TWS耳机的ANC模式

anc_poweroff

```
/*
 * @brief ANC模块关机处理函数
 *
 * 该函数在系统关机或低功耗时被调用，用于保存当前ANC状态并执行必要的清理工作。
 * 主要功能：
 * 1. 保存当前ANC状态到NVRAM（如果使能了记忆功能）
 * 2. 如果ANC当前处于开启状态，则执行淡出关闭操作
 */
void anc_poweroff(void)
{
    if (anc_hdl) {  // 检查ANC句柄是否有效
        // 记录当前ANC状态，用于调试
        user_anc_log("anc_cur_state:%s\n", anc_state_str[anc_hdl->state]);

#if ANC_INFO_SAVE_ENABLE  /* 记忆功能使能宏
                           * 1: 保存当前ANC状态到NVRAM
                           * 0: 不保存状态
                           */
        anc_info_save();  // 保存当前ANC信息到NVRAM
#endif/*ANC_INFO_SAVE_ENABLE*/

        // 检查当前ANC是否处于开启状态
        if (anc_hdl->state == ANC_STA_OPEN) {
            user_anc_log("anc_poweroff\n");
            /*
             * 设置模式为关闭状态
```

```
             * 注意：这里只是设置参数，实际的关闭操作在anc_fade中完成
             * 添加者：samson 20250428
             * 说明：待机时必须设置anc off但不能保存到NVRAM
             */
            anc_hdl->param.mode = ANC_OFF;

            // 执行淡出效果并关闭ANC
            // 参数0表示淡出到静音
            anc_fade(0);
        }
    }
}
```

关键点说明：

1. **函数功能**：
    - 该函数主要负责在系统关机或进入低功耗模式前，对ANC模块进行适当的关闭处理。
2. **ANC_INFO_SAVE_ENABLE宏**：
    - 控制是否在关机时保存ANC状态
    - 当设置为1时，会调用`anc_info_save()`保存当前状态
    - 当设置为0时，不保存状态
3. **状态处理**：
    - 只有在ANC当前处于开启状态(`ANC_STA_OPEN`)时，才需要执行关闭操作
    - 关闭时先将模式设置为`ANC_OFF`，然后调用`anc_fade(0)`执行淡出效果
4. **重要注释**：
    - 特别说明了`anc_hdl->param.mode = ANC_OFF`这行代码的作用和注意事项
    - 强调了在待机时必须设置ANC为关闭状态，但不能保存到NVRAM
5. **错误处理**：
    - 函数开始时会检查`anc_hdl`是否有效
    - 通过日志记录当前状态，便于问题排查

这个函数是ANC模块生命周期管理的重要组成部分，确保在系统关机或进入低功耗模式时，ANC模块能够正确关闭并保存必要状态。

流程说明

1. **系统启动流程**：
    - 系统上电后，在`earphone.c`中调用`anc_poweron`
    - 检查`TCFG_AUDIO_ANC_ENABLE`宏决定是否启用ANC
2. **记忆功能开启路径**（`ANC_INFO_SAVE_ENABLE=1`）：
    - 从NVRAM读取之前保存的ANC配置
    - 如果读取成功且当前状态为INIT，则恢复到之前保存的模式
    - 如果读取失败或状态不匹配，则保持当前状态
3. **记忆功能关闭路径**（`ANC_INFO_SAVE_ENABLE=0`）：
    - 使用默认配置
    - 设置模式为`ANC_OFF`
4. **运行中模式切换**：
    - 用户可以通过界面切换ANC模式
    - 如果记忆功能开启，新配置会保存到NVRAM
    - 如果记忆功能关闭，切换仅对当前会话有效
5. **系统关机流程**：
    - 系统关机时调用`anc_poweroff`
    - 如果记忆功能开启，保存当前状态到NVRAM
    - 执行ANC淡出关闭

关键点

1. **记忆功能开启时**：
    - 开机时会恢复到上次关机时的模式
    - 模式切换会实时保存到NVRAM
    - 关机时会保存当前状态
2. **记忆功能关闭时**：
    - 开机时总是使用默认配置
    - 模式切换仅在当前会话有效
    - 关机时不保存状态
3. **错误处理**：
    - 如果NVRAM读取失败，会记录错误日志

- 如果ANC句柄无效，会提前返回