AW30N

SDK培训

作者　　杰理AD应用研发组

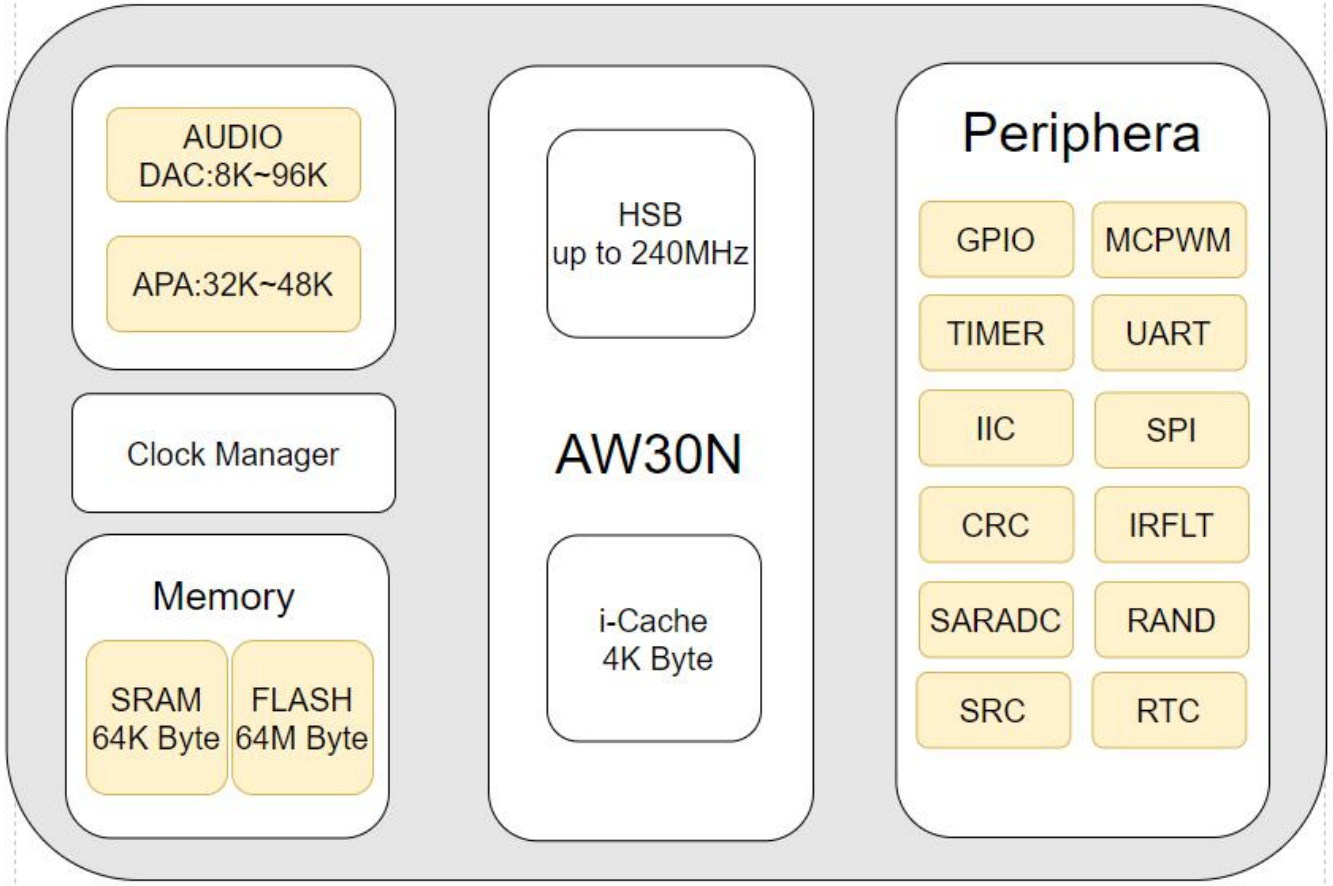
2024/1/31

目录

01 芯片规格介绍

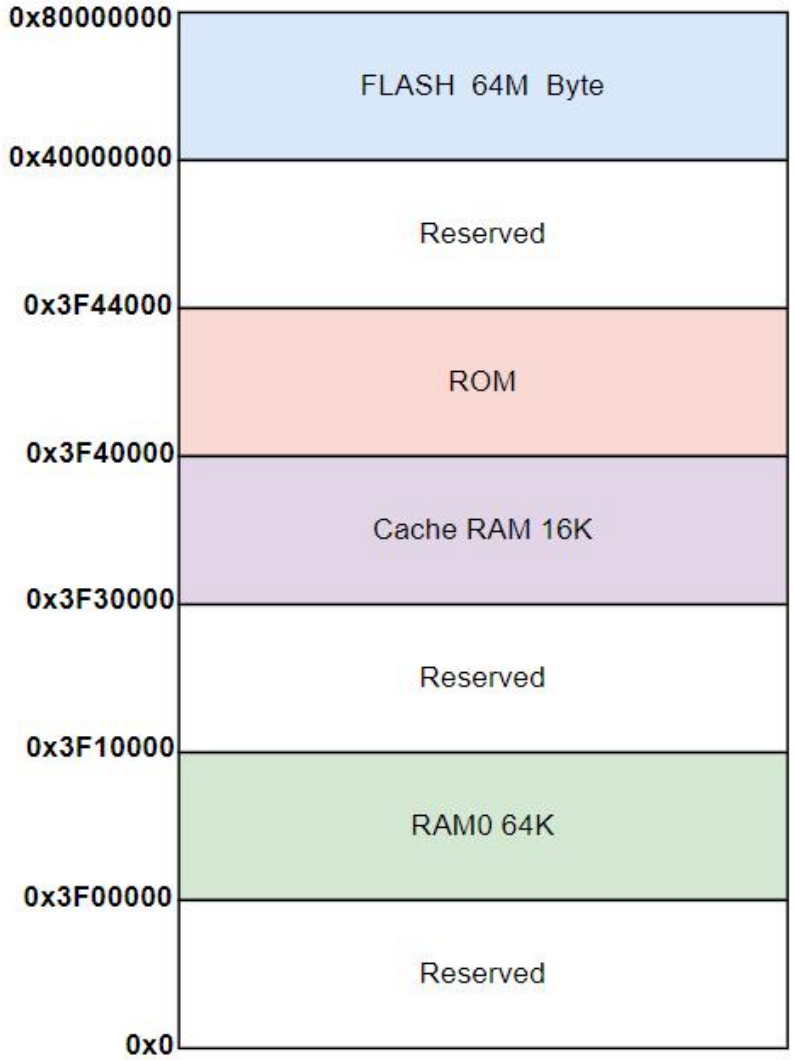
介绍AW30N芯片基本规格

一、芯片规格介绍



- **AW30N芯片规格：**
- 1、拥有单声道APA和AUDIO DAC；
- 2、HSB max 240MHz，LSB max 120MHz；
- 3、64K字节SRAM；

一、Memory内存介绍



AW30N

- **AW30N Memory 映射：**
- 1、拥有64M字节的FLASH寻址空间；
- 2、拥有64K字节SRAM；
- 3、拥有16K字节ICACHE缓存空间；

一、不同芯片差异介绍

杰理科技AD系列芯片主要差异

芯片	AD14N	AD15N	AD17N	AD18N	AD16N	AW30N
CPU	32位	32位	32位	32位	32位	32位
最高运行时钟	192MHz	160MHz	160MHz	160MHz	160MHz	240MHz
cache最大flash寻址空间	32M字节	64M字节	64M字节	64M字节	64M字节	64M字节
RAM	32K	20K	14K	40K	40K	64K
Cache ram	16K	8K	4K	16K	16K	16K
AUDIO_ADC	MIC/AUX	无	无	无	MIC/AUX	MIC/AUX
AUDIO ADC采样率	8K~24K	无	无	无	8K~48K	8K~48K
CLASS-D直驱喇叭	支持	支持	支持	支持	无	支持
APA THD	-33dB	-33dB	-70+dB	-70+db	无	-72+dB
模拟AUDIO DAC	单声道	无	无	单声道	立体声	单声道
模拟AUDIO DAC SNR	83db	无	无	81dB	>97db	>92dB
AUDIO DAC/APA采样率	8K~32K	8K~32K	32K~48K	AUDIO_DAC:8K~96K APA:32K~48K	8K~96K	AUDIO_DAC:8K~96K APA:32K~48K
gpio	25MAX	33MAX	22MAX	47MAX	33MAX	25MAX
io映射	2 out 4 input	4 out 4 input	I/O Crossbar	I/O Crossbar	I/O Crossbar	I/O Crossbar
SDMMC	支持	支持	不支持	不支持	支持	支持
USB	支持	不支持	不支持	不支持	支持	支持

杰理科技AD系列芯片主要差异--软件

芯片	AD14N	AD15N	AD17N	AD18N	AD16N	AW30N
解码格式	MP3、WAV、F1A、A、ump3、midi	F1A、A、midi	F1A、A、midi	MP3、WAV、F1A、A、ump3、midi	MP3、WAV、F1A、A、ump3、midi	MP3、WAV、F1A、A、ump3、midi、OPUS、SPEEX、ADPCM、SBC、MSBC
压缩格式	MP3、WAV、A、ump3				MP3、WAV、A、ump3	MP3、WAV、A、ump3、OPUS、SPEEX、SBC、MSBC、ADPCM
扩展多种音频算法	支持	不支持	不支持	支持	支持	支持

一、RF相关性能指标介绍

杰理科技 JIELI TECHNOLOGY 用芯美好世界 Chips Delight the World

AW30 - master

Search docs

3. RF相关性能指标

3.1. AUDIO BLE

AW30N BLE在发射功率为0dBm档位条件下，测到BLE灵敏度性能如下:

芯片BLE灵敏度	1Mbps	2Mbps	Long range(S2)	Long range(S8)
2402	-97dBm	-92dBm	-99dBm	-104dBm
2440	-97dBm	-94dBm	-98dBm	-104dBm
2448	-95dBm	-93dBm	-97dBm	-97dBm
2480	-97dBm	-94dBm	-99dBm	-104dBm

AW30N BLE在发射功率为0dBm档位条件下，测到BLE实际发射功率性能如下:

芯片BLE发射功率	1Mbps	2Mbps	Long range(S2)	Long range(S8)
2402	0.10dBm	0.07dBm	0.00dBm	0.00dBm
2440	0.40dBm	0.40dBm	0.36dBm	0.31dBm
2448	0.47dBm	0.48dBm	0.37dBm	0.34dBm
2480	0.42dBm	0.44dBm	0.32dBm	0.29dBm

一、RF蓝牙距离

测试场景		空旷场景（最大测试到60米）		
测试条件		面对面	背对背（单人遮挡）	背对背（双人遮挡）
测试样机	AW300开发板	a、轻微卡顿：40m-55m b、卡顿严重：60m c、断开连接：无	a、轻微卡顿：30m-40m b、卡顿严重：40m-60m c、断开连接：无	a、轻微卡顿：30-40米 b、卡顿严重：40m-50m c、断开连接：55米

测试场景		复杂场景（最大测试到35米）	
测试条件		面对面	背对背（单人遮挡）
测试样机	AW300开发板	a、轻微卡顿：24m b、卡顿严重：32-35m c、断开连接：无	a、轻微卡顿：16m b、卡顿严重：16m-35m c、断开连接：无

02 BLE音频传输框架与流程

介绍GATT服务，介绍音频传输代码层级、
发送接收流程

二、SDK BLE应用音频传输框架与流程-两种GATT服务（完整GATT）

完整GATT服务，在SDK中其收发接口通过vble_complete来进行管理，
详细见vble_complete.c

- **GATT_complete特点：**
- 1、基于标准GATT协议，具有完整GATT profile
- 2、从机支持HOGP协议，可连接手机或者其他主机设备
- 3、支持私有协议，支持修改广播包内容以及参数
- 4、支持多个att_handle收发数据，相关配置由profile决定
- 5、不支持主从切换

- **涉及文件：**
- vble_complete.c
- ble_hogp.c
- ble_hogp_profile.h

二、SDK BLE应用音频传输框架与流程-两种GATT服务（简易GATT）

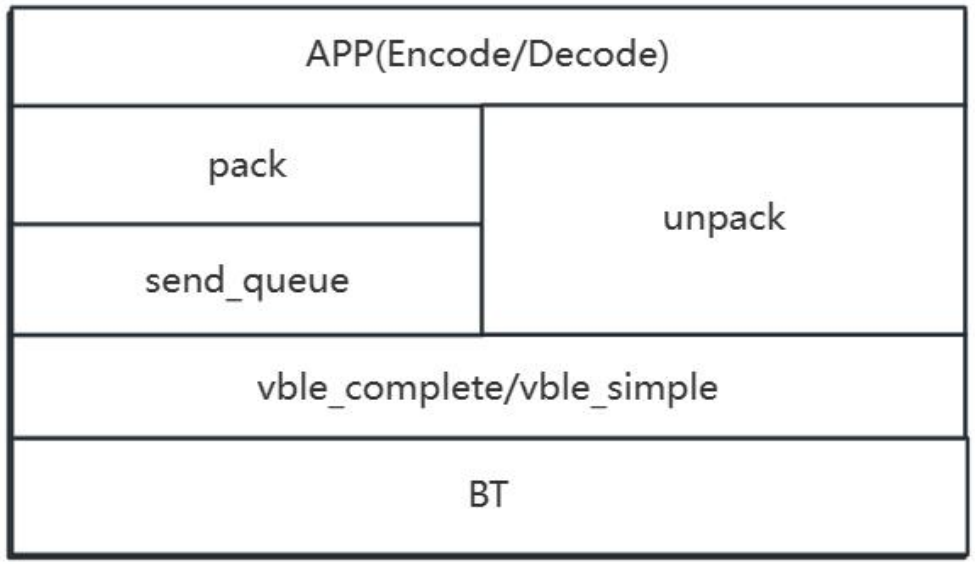
简易GATT服务，在SDK中其收发接口通过vble_simple来进行管理，详细见vble_simple.c

- **GATT_simple特点：**
- 1、具有简易的GATT profile
- 2、仅支持简单数据收发
- 3、使用固定att_handle通道进行收发数据
- 4、支持主从切换，主机可切换至从机进行蓝牙升级
- 5、支持私有连接间隔配置（支持大于3000us的任意间隔）

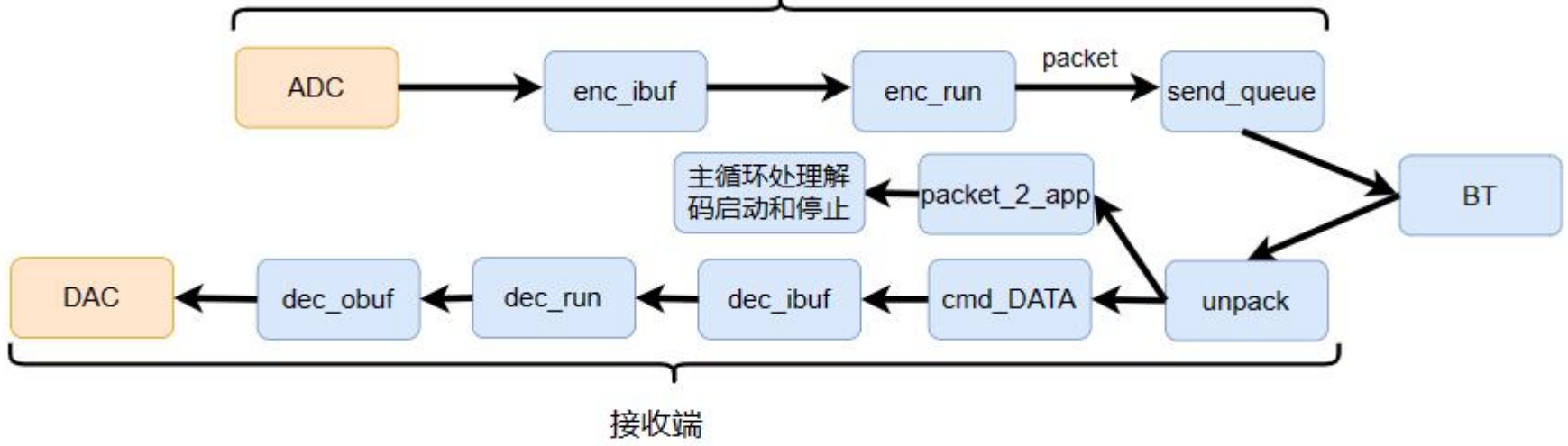
- **涉及文件：**
- vble_simple.c
- ble_master.c
- ble_slave.c

二、SDK BLE音频传输框架与流程-音频数据传输框架

aw30 音频传输代码架构

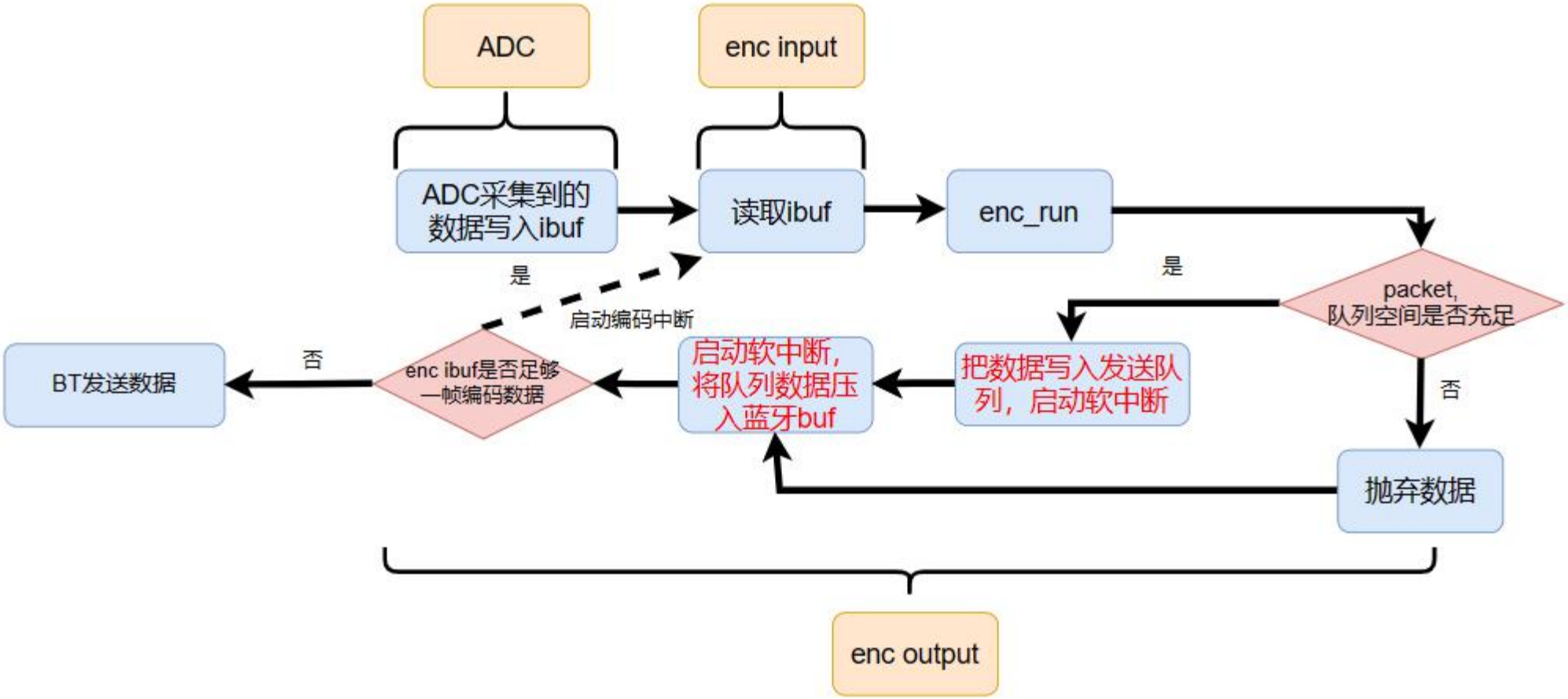


发送端

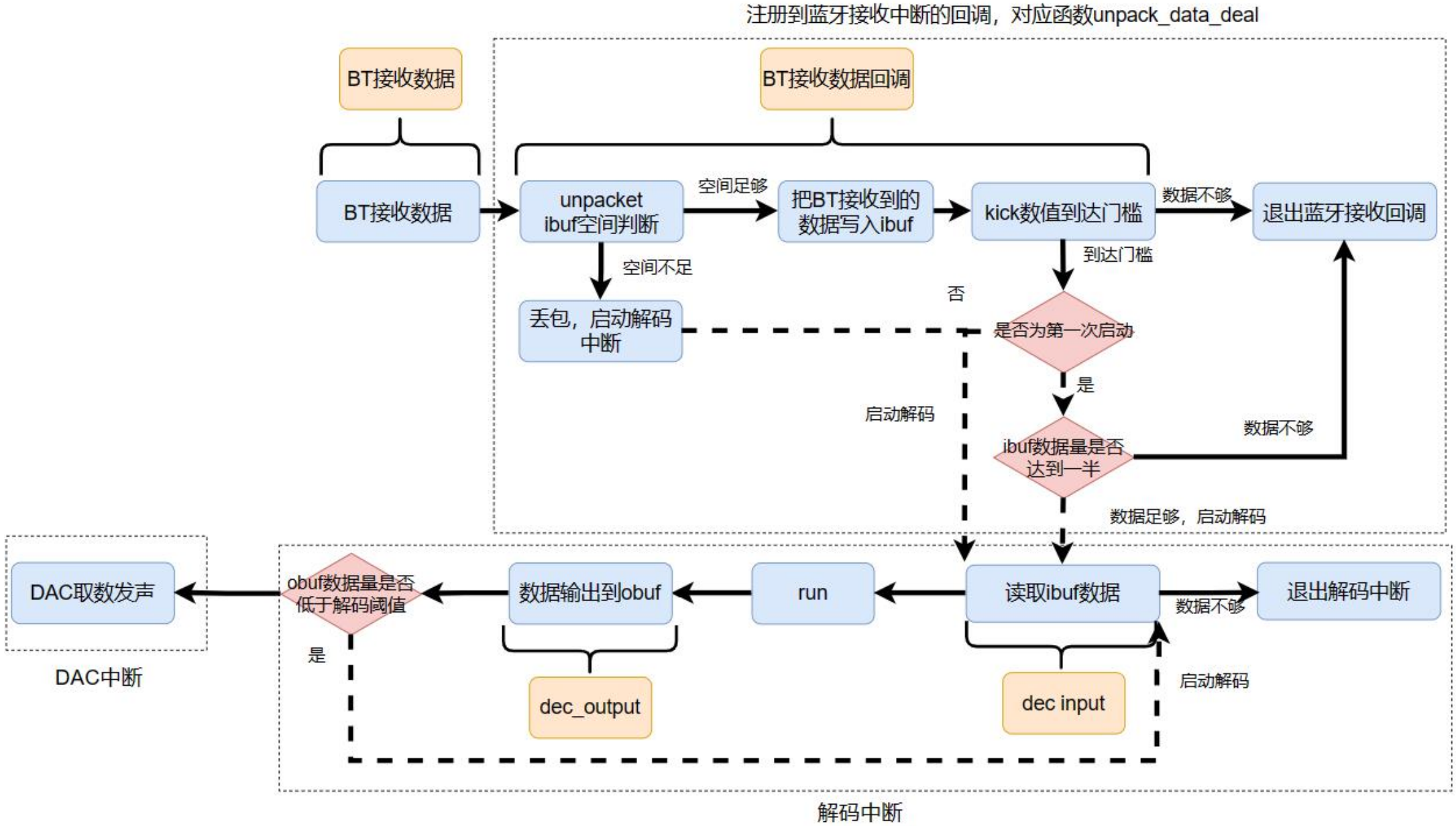


接收端

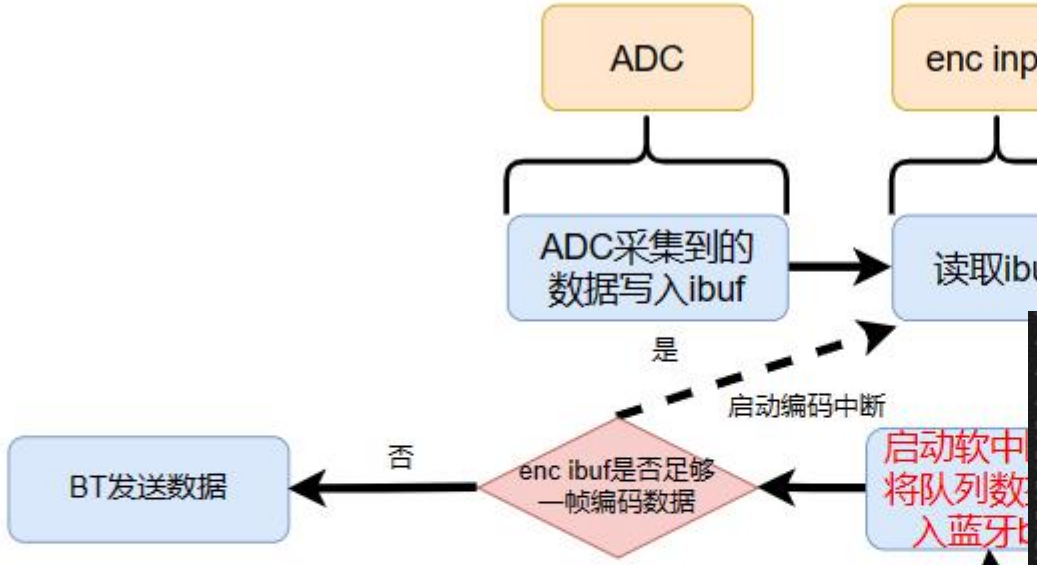
二、SDK BLE音频传输框架与流程-音频数据传输流程（发送端）



二、SDK BLE音频传输框架与流程-音频数据传输流程（接收端）



二、SDK BLE音频传输框架与流程-数据打包(packet)流程（发送端）



packet流程节点

```c
u16 ar_trans_pack(RADIO_PACKET_TYPE type, u8 *data, u16 data_len, u8 *packet_buf)
{
    u16 crc16_tmp;
    u16 slen = 0;

    RF_RADIO_PACKET packet = {0};
    packet.header       = PACKET_HEAD;
    packet.type         = type;
    packet.data_length = data_len;
    packet.packet_index = rf_send_cnt++;
    crc16_tmp = CRC16_with_initval(&(packet.type), RADIO_PACKET_CRC_LEN, 0);
    if (NULL != data) {
        crc16_tmp = CRC16_with_initval(data, data_len, crc16_tmp);
    }
    packet.crc8_l = crc16_tmp & 0xff;

    memcpy(&packet_buf[0], &packet, sizeof(packet));
    slen += sizeof(packet);
    if (NULL != data) {
        memcpy(&packet_buf[sizeof(packet)], data, data_len);
        slen += data_len;
    }
    /* log_info("S cnt:%d type:%d len:%d crc:0x%x", packet.packet_index, packet.type,
RADIO_PACKET) + packet.data_length, crc16_tmp); */
    return slen;
}
```

packet时会从type开始，
逐个字节做循环校验

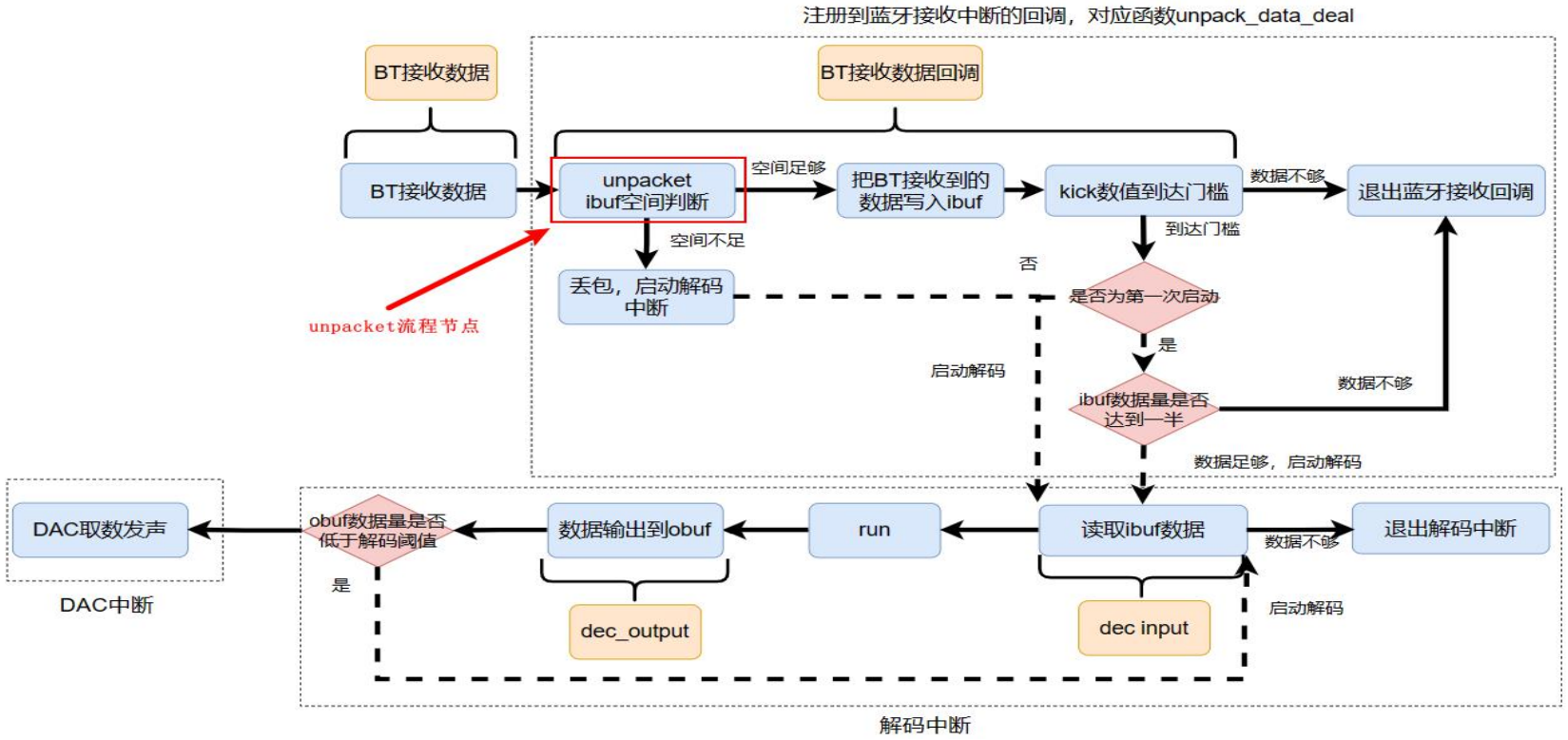
```
NORMAL  [redacted]  </bsp/common/audio_rf_trans/trans_packet.c  c  38%  33
```

```c
#define PACKET_HEAD (0xaa55)

typedef struct _RF_RADIO_PACK {
    u16 header; //0xaa55
    u8 crc8_l;
    u8 type;    //RADIO_PACKET_TYPE
    u16 data_length;
    u32 packet_index;
    u8 data[0];
} _GNU_PACKED_ RF_RADIO_PACKET;
```

packet之后的数据包结构

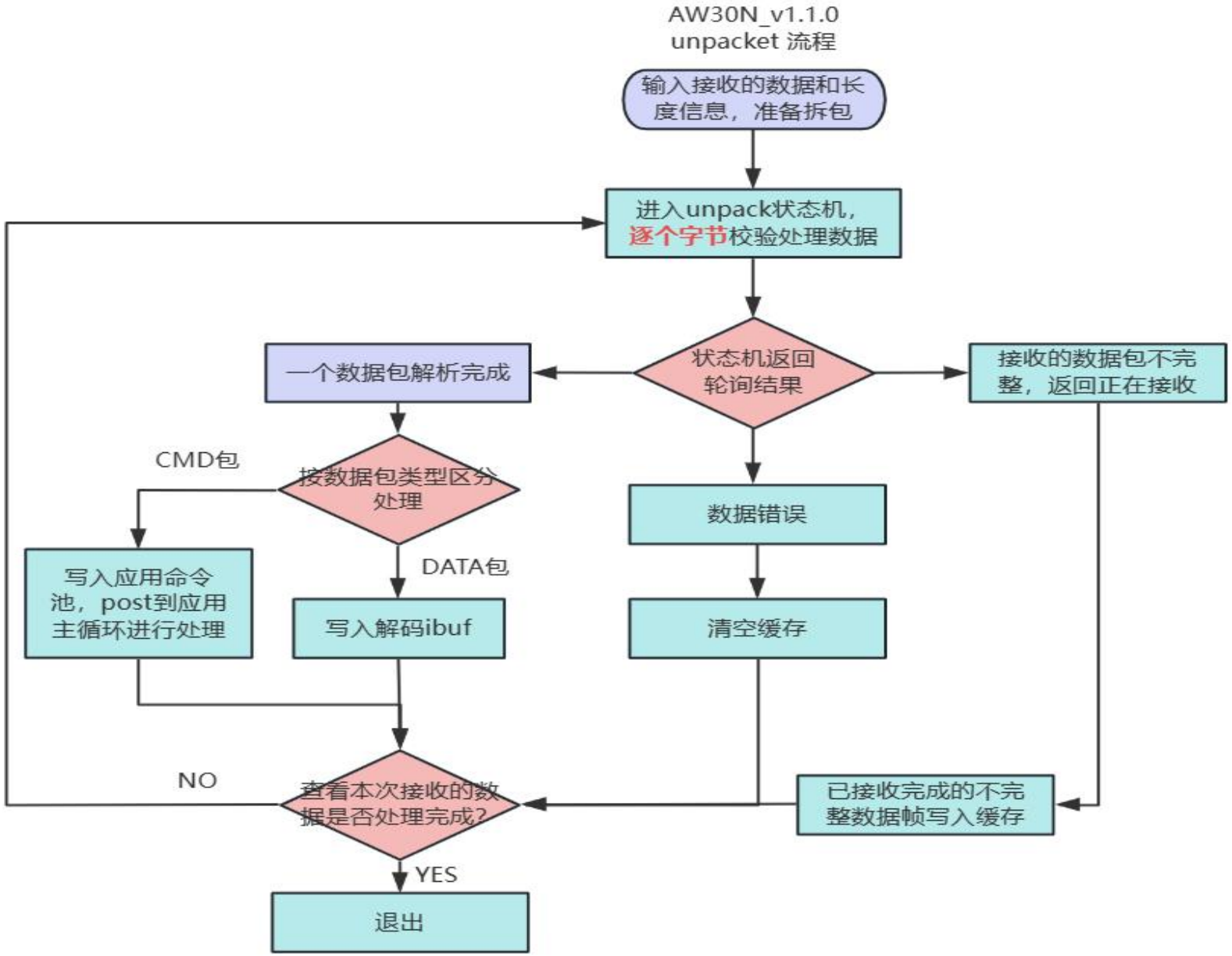
```
NORMAL  [redacted]  apps/app/bsp/common/audio_rf_trans/au
```

二、SDK BLE音频传输框架与流程-数据拆包(unpacket)流程（接收端）

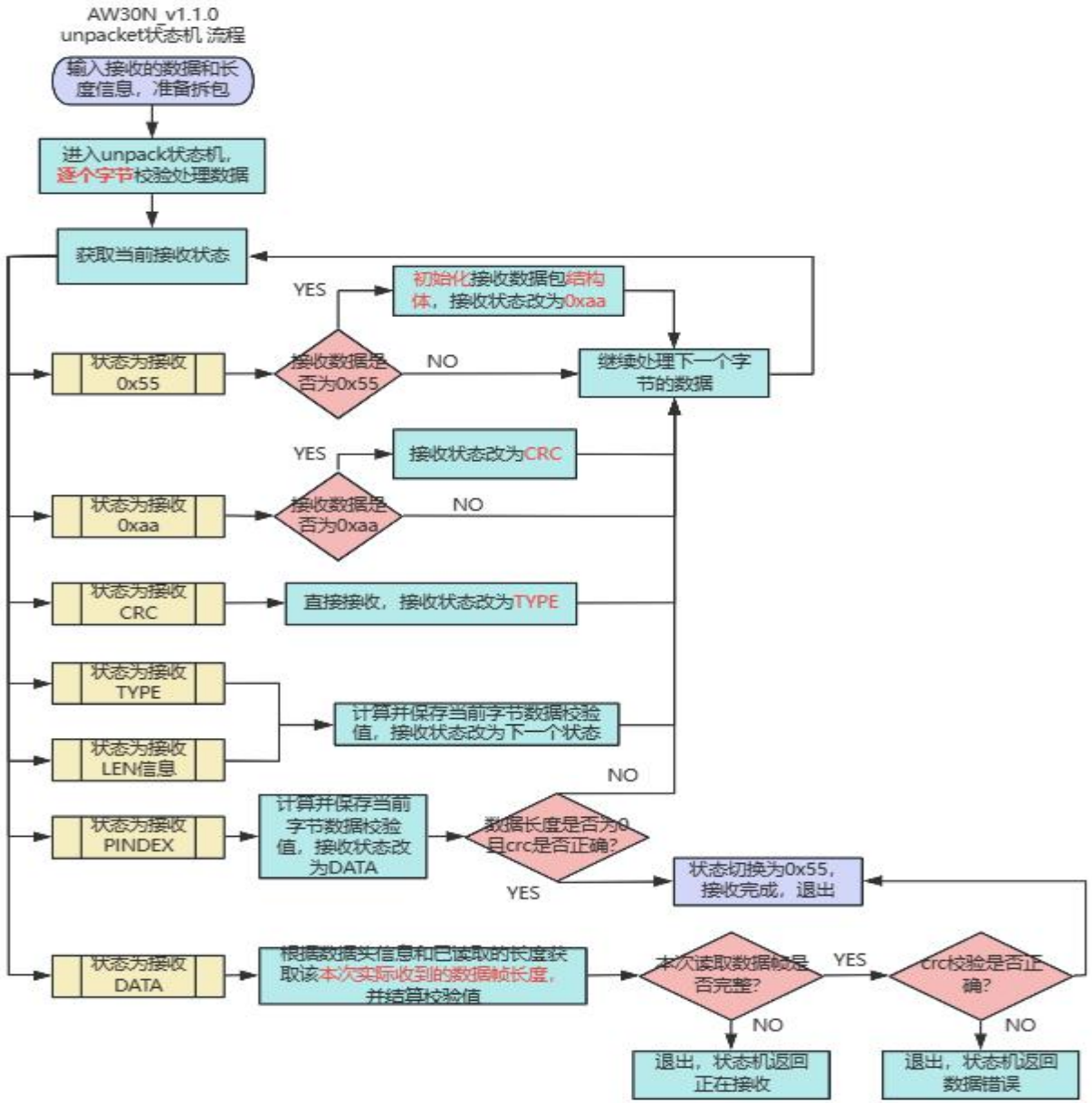


蓝牙发送数据能力有限，因此发送端压入蓝牙buf的数据包可能不完整，且接收端收到的数据会有多个数据包

二、SDK BLE音频传输框架与流程-数据拆包(unpacket)流程（接收端）



二、SDK BLE音频传输框架与流程-数据拆包(unpacket状态机)流程（接收端）



03 SDK应用框架说明

介绍各个应用框架流程

三、玩具应用

- 玩具应用流程
- 即音乐、录音、扩音、midi、AUX，RTC模式的统称
- 主要涉及文件：
  - app_modules.h
  - main.c
  - app.c
  - inita_app.c

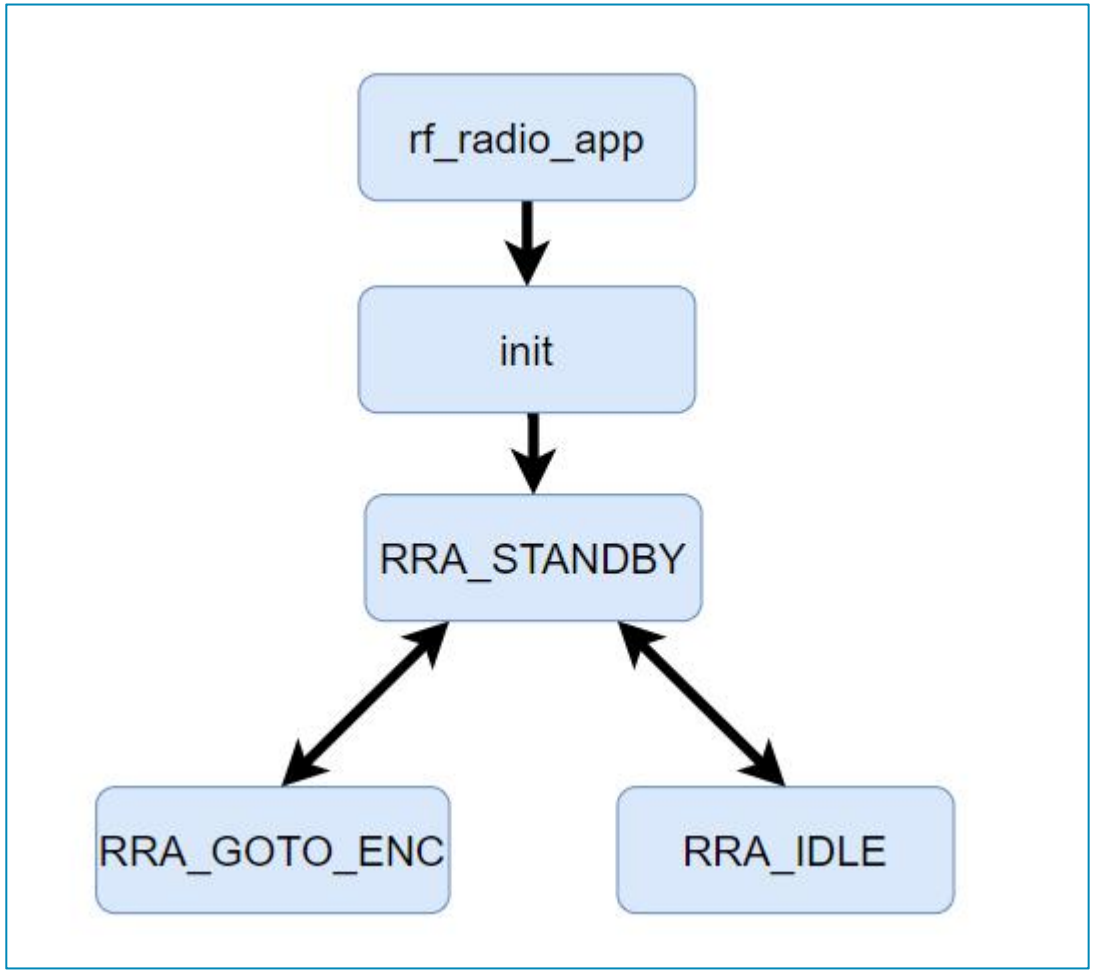
```
142 #if (0 == RF_REMOTECONTROL_MODE_EN)
143 u32 get_up_suc_flag();
144 void mbox_flash_main(void)
145 {
146     log_info("Mbox-Flash App\n");
147     if (get_up_suc_flag()) {
148         log_info("----device update end----\n");
149         wdt_close();
150         while (1);
151     }
152
153     vm_isr_response_index_register(IRQ_TICKTMR_IDX);//vm擦写flash时响应tick_
154     delay_10ms(50);//等待系统稳定
155     pa_mute(0);
156
157 #if 1//volume memory
158     u8 vol = 0;
159     u32 res = sysmem_read_api(SYSMEM_INDEX_VOL, &vol, sizeof(vol));
160     if ((vol <= 31) && (res == sizeof(vol))) {
161         dac_vol(0, vol);
162         log_info("powerup set vol : %d\n", vol);
163     }
164 #endif
165
166     sysmem_read_api(SYSMEM_INDEX_SYSMODE, &work_mode, sizeof(work_mode));
167     /* work_mode = MUSIC_MODE; */
168     /* work_mode = RECORD_MODE; */
169     /* work_mode = AUX_MODE; */
170     /* work_mode = MIDI_DEC_MODE; */
171     /* work_mode = MIDI_KEYBOARD_MODE; */
172     /* work_mode = SIMPLE_DEC_MODE; */
173     /* work_mode = REMOTECONTROL_MODE; */
174     /* work_mode = LOUDSPEAKER_MODE; */
175     /* work_mode = RTC_MODE; */
176     /* work_mode = RF_RADIO_MODE; */
177     while (1) {
178         clear_all_message();
179         //切换模式前做预擦除动作
180         sysmem_pre_erase_api();
181         switch (work_mode) {
182 #if MUSIC_MODE_EN
183         case MUSIC_MODE:
184             log_info("-Music Mode\n");
185             music_app();
NORMAL  <ps/app/src/mbox_flash_app.c   c   utf-8[unix]
```

统称玩具应用模式

三、对讲机应用

- 对讲机应用
- 主要涉及文件：
  - app_modules.h
  - rf_radio_app.c
  - rf_radio_receive.c
  - rf_radio_send.c

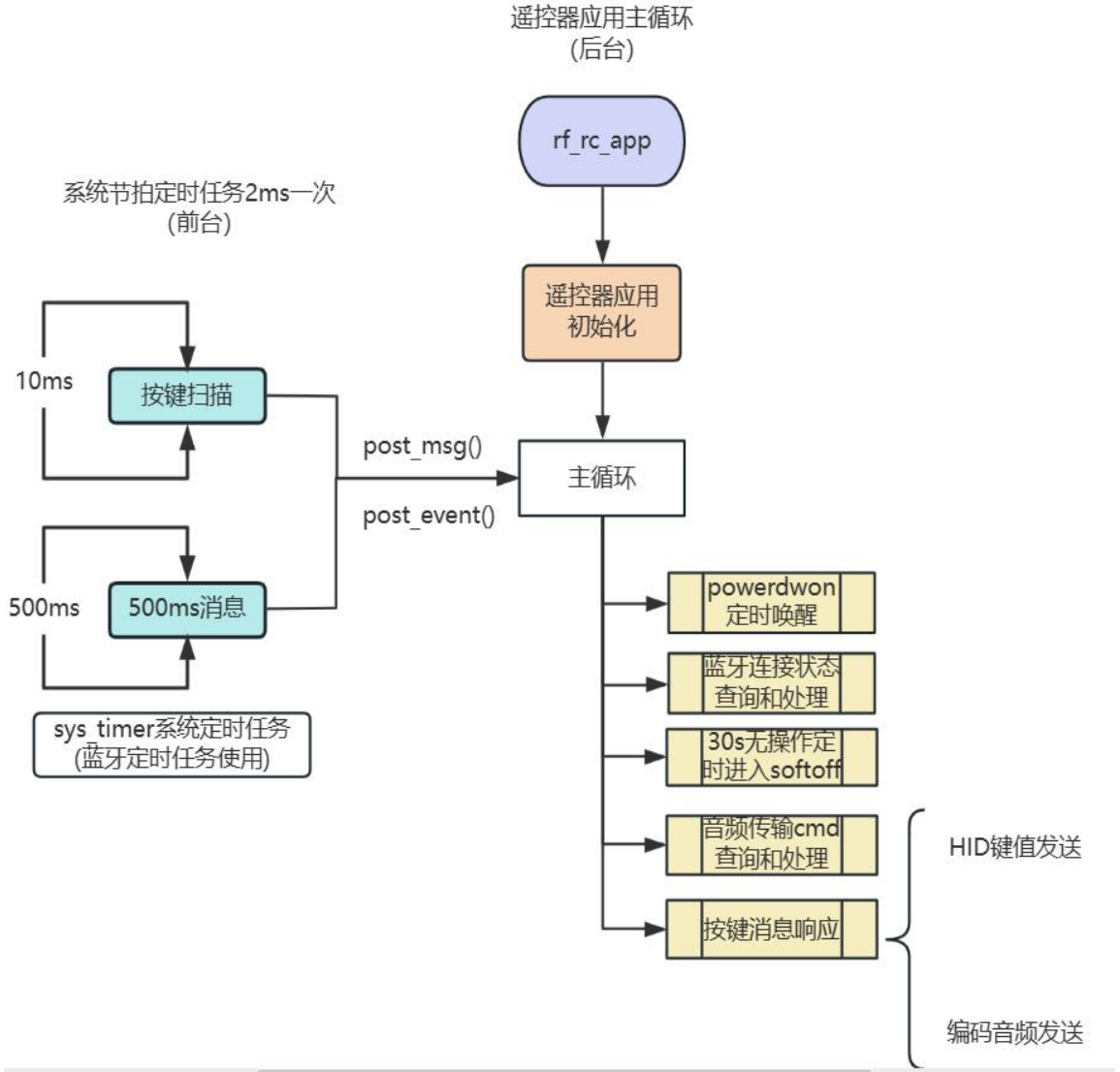
- 对讲机应用框架



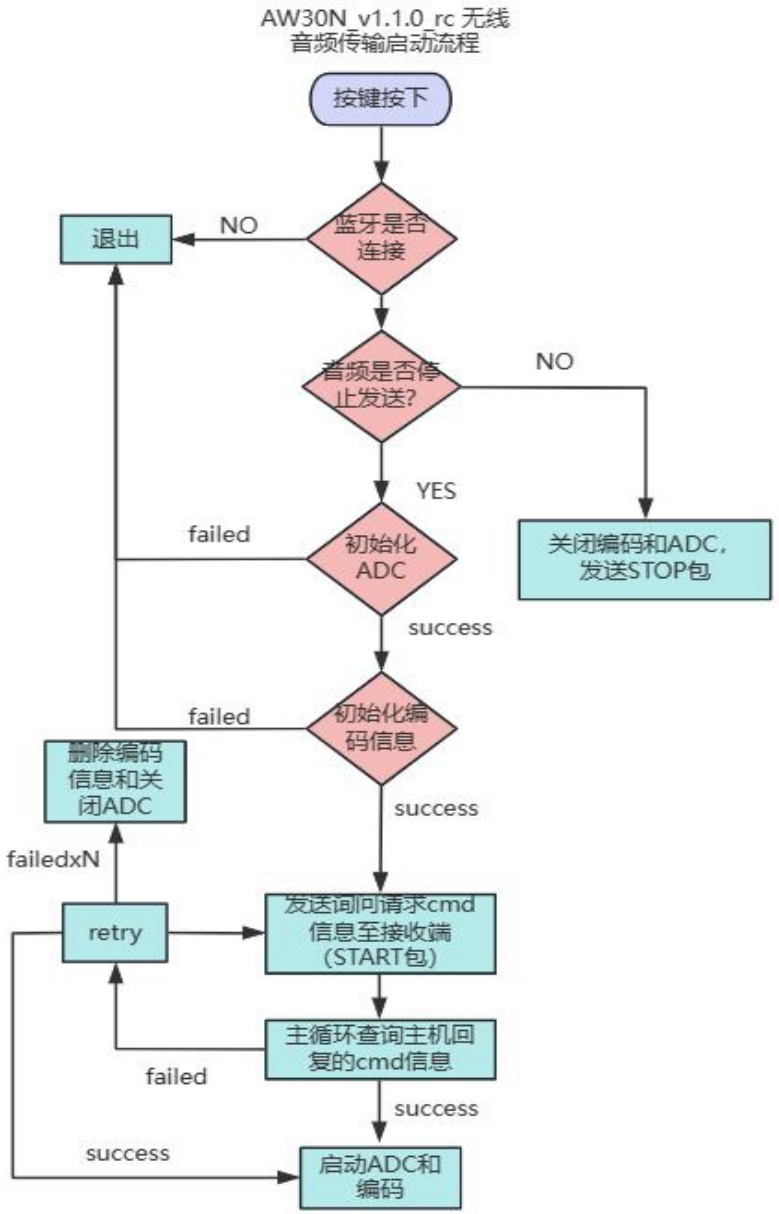
三、遥控器从机应用

- 遥控器应用
- 主要涉及文件：
  - app_modules.h
  - rc_app.c

- 遥控器从机框架



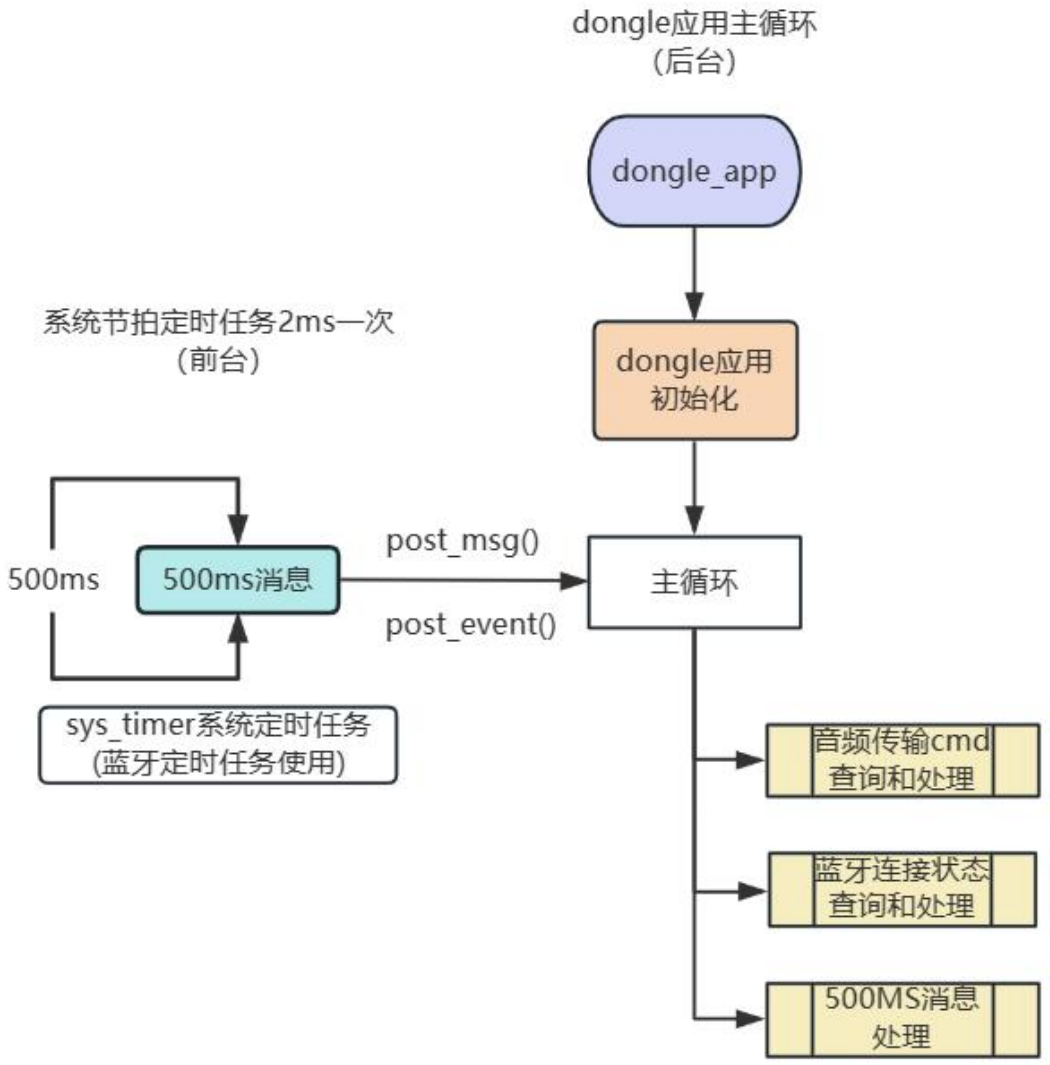
三、遥控器从机应用


AW30N_v1.1.0_rc 无线
音频传输启动流程
按键按下
蓝牙是否连接
NO
退出
音频是否停止发送?
NO
YES
关闭编码和ADC，发送STOP包
初始化ADC
failed
success
初始化编码信息
failed
success
删除编码信息和关闭ADC
failedxN
retry
发送询问请求cmd信息至接收端(START包)
主循环查询主机回复的cmd信息
failed
success
success
启动ADC和编码

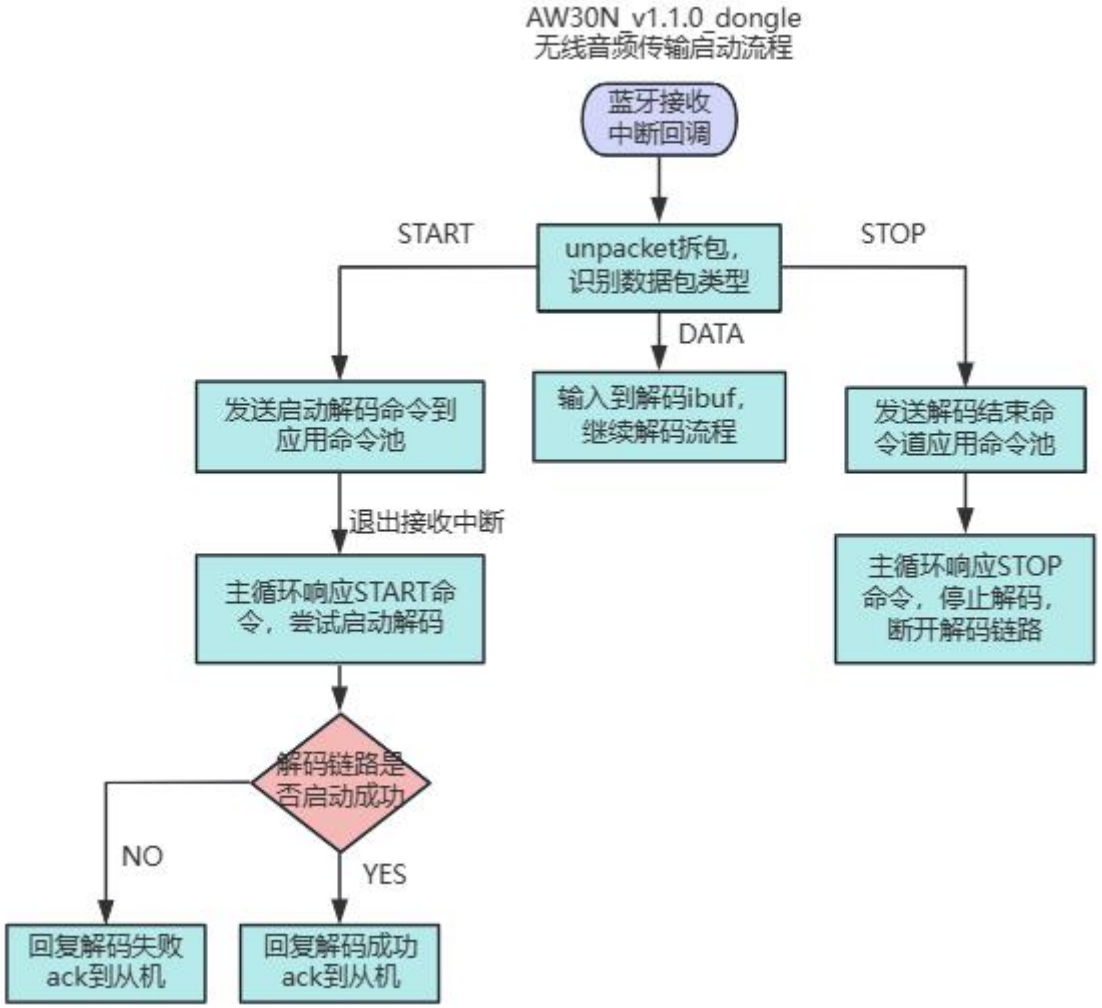
三、遥控器dongle主机应用

- 遥控器dongle应用
- 主要涉及文件：
  - app_modules.h
  - dongle.c

- 遥控器dongle框架



三、遥控器dongle主机应用



04 SDK应用配置说明

介绍各个配置文件，以及其中配置作用

四、SDK应用配置说明

- app_modules.h 部分配置说明
- （详情请看SDK文档）
- 1、应用模式选择
- 2、解码格式功能开关
- 3、音效算法功能开关
- 4、应用模式开关
- 5、升级开关

```
10 //  APP方案选择
11 #define RUN_APP_CUSTOM 1
12 #define RUN_APP_RC 0
13 #define RUN_APP_DONGLE 0
14
15
16 #if RUN_APP_CUSTOM
17
18
19 //  A格式解码
20 #define DECODER_A_EN 1
196 //  无缝循环使能
197 #define DECODER_LOOP_EN 0
198
199 //  定时任务注册功能使能
200 #define SYS_TIMER_EN 1
201
202 //  蓝牙BLE功能使能
203 #define BLE_EN 1
204
205 //  升级功能使能
206 #define UPDATE_V2_EN 1
207
208 //  充电仓/蓝牙测试盒单线串口升级
209 #define TESTBOX_UART_UPDATE_EN 0
210
211 //  测试盒蓝牙升级
212 #define TESTBOX_BT_UPDATE_EN 1
213
214 //  SD卡设备升级
215 #define SD_UPDATE_EN 1
216
217 //  U盘设备升级
218 #define UDISK_UPDATE_EN 1
219
220
221 #endif
/apps/app/src/mbox_flash/bd49/app_modules.h
```

玩具应用/对讲机应用

遥控器从机

dongle

四、SDK应用配置说明

- app_config.c 部分配置说明
- （详情请看SDK文档）
- 1、中断优先级配置
- 2、midi和midi琴配置

优先级配置

```
#include "gpio.h"
//中断优先级
//系统使用到的
const int IRQ_IRTMR_IP    = 6;
const int IRQ_AUDIO_IP    = 5;
const int IRQ_DECODER_IP  = 1;
const int IRQ_WFILE_IP    = 1;
const int IRQ_RF_QUEUE_IP   = 1;
const int IRQ_ADC_IP      = 3;
const int IRQ_ENCODER_IP  = 0;
const int IRQ_TICKTMR_IP  = 3;
const int IRQ_USB_IP      = 4;
const int IRQ_SD_IP       = 3;
const int IRQ_CTMU_IP     = 2;
const int IRQ_STREAM_IP   = 4;
const int IRQ_SPEAKER_IP  = 1;
```

midi配置

```
//内存管理malloc内部断言
const char MM_ASSERT    = TRUE;
//内核异常打印
const u8 config_asser = 1;

const u8 config_spi_code_user_cache = 1;//sfc放code区

const u8 config_audio_adc_enable = 1;

#if (DECODER_MIDI_EN || DECODER_MIDI_KEYBOARD_EN)
//midi主轨选择方式
const int MAINTRACK_USE_CHN = 0;     //0:用track号来区分  1:用channel号来区分。
const int MAX_DEC_PLAYER_CNT = 8;    //midi乐谱解码最大同时发声的key数,范围[1,31]
const int MAX_CTR_PLAYER_CNT = 15;   //midi琴最大同时发声的key数,范围[1,31]
const int NOTE_OFF_TRIGGER = 0;      //midi琴note_off time传0时，是否产生回调音符结束 1：不回调 0：回调
#endif

#if RTC_EN
const int config_rtc_enable = 1;
#else
const int config_rtc_enable = 0;
#endif
```

```
NORMAL  /apps/app/src/mbox_flash/app_config.c  c  utf-8[unix]  23%
```

四、SDK应用配置说明

- app_config.h 部分配置说明
- （详情请看SDK文档）
- 1、打印串口配置，默认使用PA5
- 2、各种按键端口配置，AD按键默认PA8
- 3、唤醒IO配置，同AD按键IO
- 4、外挂资源flash 软硬件SPI配置
- 5、USB功能开关

```
#define NO_CONFIG_PORT                          (-1)

/*---------系统时钟配置----------*/
#define TCFG_SYS_PLL_CLK                        96000000
#define TCFG_PLL_SEL                            PLL_D1p0_192M
#define TCFG_PLL_DIV                            PLL_DIV2
#define TCFG_HSB_DIV                            HSB_DIV1

/*---------Cache Configuration----------*/
#define CPU_USE_CACHE_WAY_NUMBER                4//cache_way范围:2~4

/*---------UART Configuration----------*/
#define TCFG_UART_TX_PORT                       IO_PORTA_05     //串口打印发送脚配置
// #define TCFG_UART_RX_PORT                       IO_PORTA_06     //串口打印接收脚配置
#define TCFG_UART_BAUDRATE                      1000000         //串口打印波特率配置

/*---------KEY Configuration----------*/
#define KEY_DOUBLE_CLICK_EN                     DISABLE//使能按键多击
#define KEY_IO_EN                               0//<IO按键使能
#define KEY_AD_EN                               1//<AD按键使能
#define KEY_MIC_EN                              0//<耳机按键使能
#define KEY_MATRIX_EN                           0//<矩阵按键使能
#define KEY_IR_EN                               0//<IR按键使能
#define KEY_TOUCH_EN                            0//<触摸按键使能
#define KEY_LPTOUCH_EN                          0
//AD KEY
#define AD_KEY_IO                               IO_PORTA_08
//IR KEY
#define IR_KEY_IO                               IO_PORTA_06
// #define IR_KEY_IRQ_IDX                          IRQ_TIME2_IDX
#define IR_KEY_TIMER_IDX                        3
//TOUCH KEY
#define TOUCH_KEY_IO_SEL                        {IO_PORTA_07,IO_PORTA_08}
//LPTOUCH KEY
#define LPCTMU_CH0_EN                           0//LPCTMU_CH0使能，固定引脚：PA01
#define LPCTMU_CH1_EN                           0//LPCTMU_CH1使能，固定引脚：PA02
#define LPCTMU_CH2_EN                           0//LPCTMU_CH2使能，固定引脚：PA03
#define LPCTMU_CH3_EN                           0//LPCTMU_CH3使能，固定引脚：PB00
#define LPCTMU_CH4_EN                           0//LPCTMU_CH4使能，固定引脚：PB01
#define LPCTMU_CH5_EN                           0//LPCTMU_CH5使能，固定引脚：PB02
#define LPCTMU_CH6_EN                           0//LPCTMU_CH6使能，固定引脚：PB03
#define LPCTMU_CH7_EN                           0//LPCTMU_CH7使能，固定引脚：PB04

/*---------KEY VOICE Configuration----------*/
#define D_HAS_KEY_VOICE                         DISABLE

NORMAL  apps/app/src/mbox_flash/app_config.h    cpp  utf-8[un
```

时钟配置

串口配置

按键使能

四、SDK应用配置说明

- cpu_config.c
- 部分配置说明
- （详情请看SDK文档）
- 1、MIC各项配置

```
#endif

/*************audio adc analog config*************************************/

/*************audio analog config*****************************************
 * 以下配置，作用为系统音频 模块开机默认配置
 * */
const bool config_vcm_cap_addon = 0;// 0 :vcm不会外挂电容；1：vcm会外挂电容；
u32 const config_audio_low_voltage_mode = 0;//0:高压 1:低压
u32 const audio_adc_mic_pga_n6db = 0;// 0:0dB  1:-6dB
AUDIO_MICPGA_G const audio_adc_mic_pga_g = AUMIC_21db;//后级增益
AUDIO_MIC_INPUT_MODE const audio_adc_mic_input_mode = mic_input_anal_pa4;//micin输入方式

#if AMM_RS_INSIDE_ENABLE
AUDIO_MIC_RS_MODE const audio_adc_mic_rs_mode = mic_rs_inside;
#elif AMM_RS_OUTSIDE_ENABLE
AUDIO_MIC_RS_MODE const audio_adc_mic_rs_mode = mic_rs_outside;
#else
AUDIO_MIC_RS_MODE const audio_adc_mic_rs_mode = mic_rs_null;
#endif

#if AMM_DIFF_ENABLE
u32 const audio_adc_diff_mic_mode = 1;// 差分mic使能(N端固定PA6) 0:单端mic  1:差分mic
#else
u32 const audio_adc_diff_mic_mode = 0;// 差分mic使能(N端固定PA6) 0:单端mic  1:差分mic
#endif

AUDIO_MICBIAS_RS const audio_adc_mic_bias_rs = AUMIC_1k5;//micbias内部偏置电阻选择
AUDIO_MICLDO_VS const audio_adc_mic_ldo_vs = AUMIC_2v0;//micldo偏置电压选择

u32 const audio_adc_diff_aux_mode = 0;// 差分aux使能(N端固定PA6) 0:单端aux  1:差分aux
AUDIO_MICPGA_G const audio_adc_aux_pga_g = AUMIC_15db;//linein输入增益
AUDIO_MIC_INPUT_MODE const audio_adc_aux_input_mode = mic_input_anal_pa4;//auxin输入方式

u32 const audio_adc_con1 = A1_ADC_ASYNC_IE | A1_ADC_HPSET | A1_ADC_DC_ENABLE;

//////////////////////update/////////////////////////////
#define CONFIG_DOUBLE_BANK_ENABLE 0
#define OTA_TWS_SAME_TIME_NEW 0
/* #ifdef CONFIG_256K_FLASH */
NORMAL  apps/app/src/mbox_flash/bd49/cpu_config.c        c  utf-8[unix]  63% 部
```

四、SDK应用配置说明 - 不同模式下按键消息响应

```
static void music_idle_deal(void)
{
    OS_ENTER_CRITICAL();
    pa_mute(1);
    LED5X7_init();
    sys_power_down(-2);
    OS_EXIT_CRITICAL();

    /* powerdown应用恢复 */
    if (usb_otg_online(0) == HOST_MODE) {
        usb_host_resume(0);
        usb_write_faddr(0, 8);
    }

    void dac_power_on(u32 sr);
    dac_power_on(SR_DEFAULT);
    pa_mute(0);
}
#endif

static void music_info_init(u8 *p_dev)
{
#if KEY_IR_EN                          举例：音乐模式下的按键注册
    Sys_IRInput = 1;
#endif
    key_table_sel(music_key_msg_filter);
    decoder_init();
    dac_fade_out_api();
    err_device = 0;
    fsn_music = 0;
    err_device = 0;

    memset(&pctl[0], 0, sizeof(pctl));
    memset(&dev_scan_info[0], 0, sizeof(dev_scan_info));
    memset(&breakpoint[0], 0, sizeof(breakpoint));

    pctl[0].pdp = &breakpoint[0];
    pctl[0].dev_index = NO_DEVICE;
    pctl[0].dec_type = BIT_WAV | BIT_MP3_ST | BIT_F1A1 | BIT_A | BIT_UMP3;  //播放需要使用的解码器
#if HAS_SYDFS_EN
    pctl[0].pdir = (void *)&dir_inr_tab[0];
    pctl[0].dir_index = 0;
#endif
    u8 device;
    if (sizeof(u8) == sysmem_read_api(SYSMEM_INDEX_ACTIVE_DEV, &device, sizeof(u8))) {
        *p_dev = device;
    } else {
        *p_dev = 0;
    }
}
int decoder_eq_mode_switch(dec_obj *obj)
{
    if ((obj == NULL) || (obj->eq == NULL)) {
        return -1;
```

```
#endif

#define ADKEY_MUSIC_SHORT \
                                /*00*/      NO_MSG,\
                                /*01*/      NO_MSG,\
短按第几个按键推出对应消息      /*02*/      NO_MSG,\
                                /*03*/      NO_MSG,\
                                /*04*/      NO_MSG,\
                                /*05*/      NO_MSG,\
                                /*06*/      NO_MSG,\
                                /*07*/      NO_MSG,\
                                /*08*/      NO_MSG,\
                                /*09*/      NO_MSG,\

const u16 adkey_msg_mbox_music_table[][AD_KEY_MAX_NUM] = {
    /*短按*/        {ADKEY_MUSIC_SHORT},
    /*短按抬起*/    {ADKEY_MUSIC_SHORT_UP},
    /*长按*/        {ADKEY_MUSIC_LONG},
    /*连按*/        {ADKEY_MUSIC_HOLD},
    /*长按抬起*/    {ADKEY_MUSIC_LONG_UP},
#if (KEY_DOUBLE_CLICK_EN)
    /*双击*/        {ADKEY_MUSIC_DOUBLE_KICK},
    /*三击*/        {ADKEY_MUSIC_TRIPLE_KICK},
#endif
};
#endif

u16 music_key_msg_filter(u8 key_status, u8 key_num, u8 key_type)
{
    u16 msg = NO_MSG;
    switch (key_type) {
#if KEY_IO_EN                          IO按键类型
    case KEY_TYPE_IO:
        msg = iokey_msg_mbox_music_table[key_status][key_num];
        break;
#endif
#if KEY_AD_EN                          AD按键类型
    case KEY_TYPE_AD:
        msg = adkey_msg_mbox_music_table[key_status][key_num];
        break;
#endif
#if KEY_IR_EN                          红外按键类型
    case KEY_TYPE_IR:
        msg = irff00_msg_mbox_music_table[key_status][key_num];
        break;
#endif
    default:
        break;
    }

    return msg;
}
```

NORMAL </app/src/mbox_flash/music/music_key_table.c

四、SDK应用配置说明 - 蓝牙发射功率配置说明

2. 蓝牙硬件相关配置

2.1. 蓝牙发射功率

发射功率由函数void bt_max_pwr_set(u8 pwr, u8 pg_pwr, u8 iq_pwr, u8 ble_pwr)的参数ble_pwr配置;

其中，发射功率档位对应功率如下表:

支持的发射功率档位	对应的发射功率
0	-24dBm档位
1	-16dBm档位
2	-12dBm档位
3	-8dBm档位
4	-4dBm档位
5（默认值）	+0dBm档位
6	+6dBm档位

目前可支持档位

暂不支持

```
#define ble_clock_init()                                        \
        SFR(JL_CLOCK->PRP_CON2, 0, 2, 1); \
        SFR(JL_CLOCK->PRP_CON2, 2, 2, 2); \
        SFR(JL_CLOCK->PRP_CON2, 4, 2, 2); \
        SFR(JL_CLOCK->PRP_CON2, 8, 1, 1); \
        asm("csync"); \
        bt_pll_para(48000000, 48000000, 0, 0); \
        extern u8 bt_get_pwr_max_level(void); \
        bt_max_pwr_set(10, 5, 8, bt_get_pwr_max_level());

apps/app/bsp/common/bt_common/bt_ble.h
```

发射功率挡位配置
bt_get_pwr_max_level()默认为0dBm

蓝牙初始化发射功率

四、SDK应用配置说明 - 蓝牙相关配置说明

- 简易GATT
  - 主从机蓝牙名字
  - 连接配对名
  - 蓝牙连接间隔配置
  - 蓝牙窗口配置
  - 蓝牙超时配置
  - 蓝牙连接latency配置
- 完整GATT
  - 主从机蓝牙名字
  - 连接配对名
  - 蓝牙连接间隔配置
  - 蓝牙窗口配置
  - 蓝牙超时配置
  - 蓝牙连接latency配置

四、SDK应用配置说明 - 蓝牙相关配置说明

```
    /* log_info_hexdump(report_data->blob, report_data->blob_length); */

    switch (report_data->packet_type) {
    case GATT_EVENT_NOTIFICATION:  //notify
        break;
    case GATT_EVENT_INDICATION://indicate
    case GATT_EVENT_CHARACTERISTIC_VALUE_QUERY_RESULT://read
        break;
    case GATT_EVENT_LONG_CHARACTERISTIC_VALUE_QUERY_RESULT://read long
        break;

    default:
        break;
    }
}

static const u8 test_remoter_name1[] = "BD49_BLE(BLE)";//
/* static const u8 test_remoter_name2[]    "AC630N_HID567(BLE)";// */
static u16 default_client_write_handle;
static u16 test_client_timer = 0;

static const client_match_cfg_t match_dev01 = {
    .create_conn_mode = BIT(CLI_CREAT_BY_NAME),
    .compare_data_len = sizeof(test_remoter_name1) - 1, //去结束符
    .compare_data = test_remoter_name1,
    .bonding_flag = 0,
};

/* static const client_match_cfg_t match_dev02 = { */
/* .create_conn_mode = BIT(CLI_CREAT_BY_NAME), */
/* .compare_data_len = sizeof(test_remoter_name2) - 1, //去结束符 */
/* .compare_data = test_remoter_name2, */
/* .bonding_flag = 1, */
/* }; */

static void default_test_write(void)
{
    static u8 count[300];
    int ret = client_operation_send(default_client_write_handle, count, 300, ATT_OP_WRITE_WITHOUT_RESPOND);
    log_info("test_write:%x", ret);
}

static void default_event_callback(le_client_event_e event, u8 *packet, int size)
{
    switch (event) {
```

配对名

```
NORMAL  [redacted] </common/bt_common/ble_test/ble_master.c[+]  c  utf-8[unix]  10% ㏑ 175/1679≡ : 14
change; before #1  2024/01/27 20:35:19
```

```
#include "log.h"

#if (TESTE_BLE_EN)

BT_CONFIG bt_cfg = {
    .edr_name         = "BD49_BLE",
    .mac_addr         = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
    .tws_local_addr   = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
    .rf_power         = 10,
    .dac_analog_gain = 25,
    .mic_analog_gain = 7,
    .tws_device_indicate = 0x6688,
};

const char *bt_get_local_name()
{
    return bt_cfg.edr_name;
}

extern void bt_ble_init(void);
extern void btstack_task(void);
extern void vPortSuppressTicksAndSleep(u32 usec);
extern void wdt_clear(void);
extern void user_sele_dut_mode(bool set);


void testbox_update_msg_handle(int msg);
void ble_module(void)
{
    static u32 i = 0;
    bool temp = 0;

    SFR(JL_CLOCK->PRP_CON1,  0, 3, 1);
    SFR(JL_CLOCK->PRP_CON1, 26, 1, 1);
    SFR(JL_CLOCK->PRP_CON2, 0, 2, 1);   // wl_and_clk 1:std_48m
    SFR(JL_CLOCK->PRP_CON2, 2, 2, 2);   // wl2adc_clk 1:std_48m 2:pll_96m
    SFR(JL_CLOCK->PRP_CON2, 4, 2, 2);   // wl2dac_clk 1:std_48m 2:pll_96m
    SFR(JL_CLOCK->PRP_CON2, 8, 1, 1);   // wl_aud_rst
    asm("csync");

    bt_pll_para(48000000, 48000000, 0, 0);

#if BLE_DUT_TEST
    user_sele_dut_mode(1);
#endif
```

配置从机名

```
apps/app/bsp/common/bt_common/ble_test/bt_ble.c[+]
```

四、SDK应用配置说明 - 蓝牙相关配置说明

四、SDK应用配置说明 - 蓝牙相关配置说明

```
//---------------
// 广播周期 (unit:0.625ms)
#define ADV_INTERVAL_MIN          (160)

#define HOLD_LATENCY_CNT_MIN  (3)  //(0~0xffff)
#define HOLD_LATENCY_CNT_MAX  (20) //(0~0xffff)
#define HOLD_LATENCY_CNT_ALL  (0xffff)

static volatile hci_con_handle_t con_handle;

//加密设置
/* static const uint8_t sm_min_key_size = 7; */

//连接参数更新请求设置
//是否使能参数请求更新,0--disable, 1--enable
static const uint8_t connection_update_enable = 0; ///0--disable, 1--enable
//当前请求的参数表index
static uint8_t connection_update_cnt = 0; //

//参数表
static struct conn_update_param_t slv_conn_param;
static const struct conn_update_param_t connection_param_table[] = {
    /* {2, 2, 0, 600},//11 */
    {4, 4, 0, 100},//3.7
    {8,  20, 10, 600},
    {12, 28, 4, 600},//3.7
    {12, 24, 30, 600},//3.05
};
#define SLV_UPDATE_WAITING  BIT(0)
static volatile u8 conn_parm_update_flag;

//共可用的参数组数
#define CONN_PARAM_TABLE_CNT      (sizeof(connection_param_table)/sizeof(struct conn_update_param_t))

#if (ATT_RAM_BUFSIZE < 64)
#error "adv_data & rsp_data buffer error!!!!!!!!!!!!"
#endif

//用户可配对的，这是样机跟客户开发的app配对的秘钥
/* const u8 link_key_data[16] = {0x06, 0x77, 0x5f, 0x87, 0x91, 0x8d, 0xd4, 0x23, 0x00, 0x5d, 0xf1, 0xd8, 0xcf,
0x0c, 0x14, 0x2b}; */
#define EIR_TAG_STRING   0xd6, 0x05, 0x08, 0x00, 'J', 'L', 'A', 'I', 'S', 'D','K'
NORMAL  [redacted]  </bsp/common/bt_common/ble_test/ble_slave.c   c  utf-8[unix]  7% ≡ 101/1331≡ :  1
```

```
#else
#define ATT_LOCAL_MTU_SIZE        (512)/*(64*2)*/                    //note: need >
#define ATT_SEND_CBUF_SIZE        (1024)/*(512)*/                   //note: need >=
#endif

#define ATT_RAM_BUFSIZE           (ATT_CTRL_BLOCK_SIZE + ATT_LOCAL_MTU_SIZE + ATT_S
    //note:
static u8 att_ram_buffer[ATT_RAM_BUFSIZE] __attribute__((aligned(4)));

#define SEARCH_PROFILE_BUFSIZE    (512)                   //note:
static u8 search_ram_buffer[SEARCH_PROFILE_BUFSIZE] __attribute__((aligned(4)));
#define scan_buffer search_ram_buffer
//---------------
//搜索类型
#define SET_SCAN_TYPE       SCAN_ACTIVE
//搜索 周期大小
#define SET_SCAN_INTERVAL   26 //(unit:0.625ms)
//搜索 窗口大小
#define SET_SCAN_WINDOW     15 //(unit:0.625ms)

//连接周期
//interval >= 3000配置为私有连接参数,interval=3000,实际连接间隔为3000us
//interval <  600 配置为标准连接参数,interval=30,实际连接间隔为30*1.25ms=37.5ms
#define SET_CONN_INTERVAL   50000//(unit:1.25ms)
//连接latency
#define SET_CONN_LATENCY    0  //(unit:conn_interval)
//连接超时
#define SET_CONN_TIMEOUT    100 //(unit:10ms)
//创建连接的过程超时时间
#define INIT_CONN_TIMEOUT    2000 //(unit:ms)

//------------------------------------------------------------------------------
static u8 scan_ctrl_en;
static u8 ble_work_state = 0;
static void *app_recieve_priv = NULL;
static int (*app_recieve_callback)(void *priv, u8 *buf, u16 len) = NULL;
static void (*app_ble_state_callback)(void *priv, ble_state_e state) = NULL;
static void (*ble_resume_send_wakeup)(void) = NULL;
static u32 channel_priv;

static char gap_device_name[BT_NAME_LEN_MAX] = "jl_kkk_test";
static u8 gap_device_name_len = 0; //名字长度，不包含结束符
static hci_con_handle_t con_handle;
apps/app/bsp/common/bt_common/ble_test/ble_master.c                    c  utf-8[un
```

四、SDK应用配置说明 - 蓝牙连接间隔说明

- 简易GATT
  - 主从机蓝牙名字
  - 连接配对名
  - 蓝牙连接间隔配置
  - 蓝牙窗口配置
  - 蓝牙超时配置
  - 蓝牙连接latency配置
- 完整GATT
  - 主从机蓝牙名字
  - 连接配对名
  - 蓝牙连接间隔配置
  - 蓝牙窗口配置
  - 蓝牙超时配置
  - 蓝牙连接latency配置

四、SDK应用配置说明 - 蓝牙相关配置说明

从机蓝牙名

```
extern APP_VAR app_var;

BT_CONFIG bt_cfg = {
    .edr_name          = "JL_HID_SLAVE",
    .mac_addr          = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
    .tws_local_addr    = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
    .rf_power          = 10,
    .dac_analog_gain   = 25,
    .mic_analog_gain   = 7,
    .tws_device_indicate = 0x6688,
};

static const char edr_ext_name[] = " 3.0";

u8 get_max_sys_vol(void)
{
#if 0//TCFG_APP_FM_EMITTER_EN
N… <common/bt_common/hid/modules/user_cfg.c   c   0%   3/408≡ : 21
```

主机蓝牙名

```
extern APP_VAR app_var;

BT_CONFIG bt_cfg = {
    .edr_name          = "JL_HID_MASTER",
    .mac_addr          = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
    .tws_local_addr    = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
    .rf_power          = 10,
    .dac_analog_gain   = 25,
    .mic_analog_gain   = 7,
    .tws_device_indicate = 0x6688,
};

u8 get_max_sys_vol(void)
{
#if 0//TCFG_APP_FM_EMITTER_EN
    return FM_EMITTER_MAX_VOL;
#else
<common/bt_common/spp_and_le/modules/user_cfg.c   c   9%
```

```
static const u8 dg_test_remoter_name1[] = "Lenovo Go Multi-Device Mouse
;//键盘
/* static const u8 dg_test_remoter_name2[] = "AC897N_MX(BLE)";//鼠标 */
static const u8 dg_test_remoter_name2[] = "JL_HID_SLAVE(BLE)" //鼠标
static const u8 user_config_tag_string[] = "abc123";
static const client_match_cfg_t dg_match_device_table[] = {
    {
        .create_conn_mode = BIT(CLI_CREAT_BY_NAME),
        .compare_data_len = sizeof(dg_test_remoter_name1) - 1, //去结束
符
        .compare_data = dg_test_remoter_name1,
N… <nd_le/examples/dongle/ble_dg_central.c   c   18%   237/1314≡ : 34
```

四、SDK应用配置说明 - 蓝牙相关配置说明

```
x0c, 0x14, 0x2b};

#if CONFIG_BLE_HIGH_SPEED
//ATT发送的包长,    note: 23 <=need >= MTU
#define ATT_LOCAL_MTU_SIZE        (247)
//ATT缓存的buffer支持缓存数据包个数
#define ATT_PACKET_NUMS_MAX       (3)
#else
#define ATT_LOCAL_MTU_SIZE        (64)
//ATT缓存的buffer支持缓存数据包个数
#define ATT_PACKET_NUMS_MAX       (2)
#endif

//ATT缓存的buffer大小,  note: need >= 23,可修改
#define ATT_SEND_CBUF_SIZE        (ATT_PACKET_NUMS_MAX * (ATT_PACKET_HEAD_SIZE + ATT_LOCAL_MTU_SIZE))

static volatile hci_con_handle_t hogp_con_handle;
int ble_hid_timer_handle = 0;

//是否使能参数请求更新,0--disable, 1--enable
static uint8_t hogp_connection_update_enable = 0;

//连接参数表,按顺序优先请求,主机接受了就中止
static const struct conn_update_param_t Peripheral_Preferred_Connection_Parameters[] = {
    {6, 9,  100, 600}, //android
    /* {7, 7,  20, 300}, //mosue */
    /* {20, 20,  20, 300}, //kb */
    {12, 12, 30, 300}, //ios
    {6,  12, 30, 400},// ios fast
};
//共可用的参数组数
#define CONN_PARAM_TABLE_CNT      (sizeof(Peripheral_Preferred_Connection_Parameters)/sizeof(struct conn_update
_param_t))

static u8 hogp_adv_data[ADV_RSP_PACKET_MAX];//max is 31
static u8 hogp_scan_rsp_data[ADV_RSP_PACKET_MAX];//max is 31

static u8 first_pair_flag;     //第一次配对标记
static u8 is_hogp_active = 0;  //不可进入sleep状态
static adv_cfg_t hogp_server_adv_config;

/*-------------------------------------------*/
//普通未连接广播周期 (unit:0.625ms)
#define ADV_INTERVAL_MIN                (160 * 5)

/*-----------------------------
pps/app/bsp/common/bt_common/hid/modules/btble_hogp.c                    c  utf-8[unix]   5% ☰ 105/1951≡ :  1
pps/app/bsp/common/bt_common/spp_and_le/examples/dongle/ble_dg_central.c"
```

完整GATT dongle连接间隔，latency，timeout

请求使能

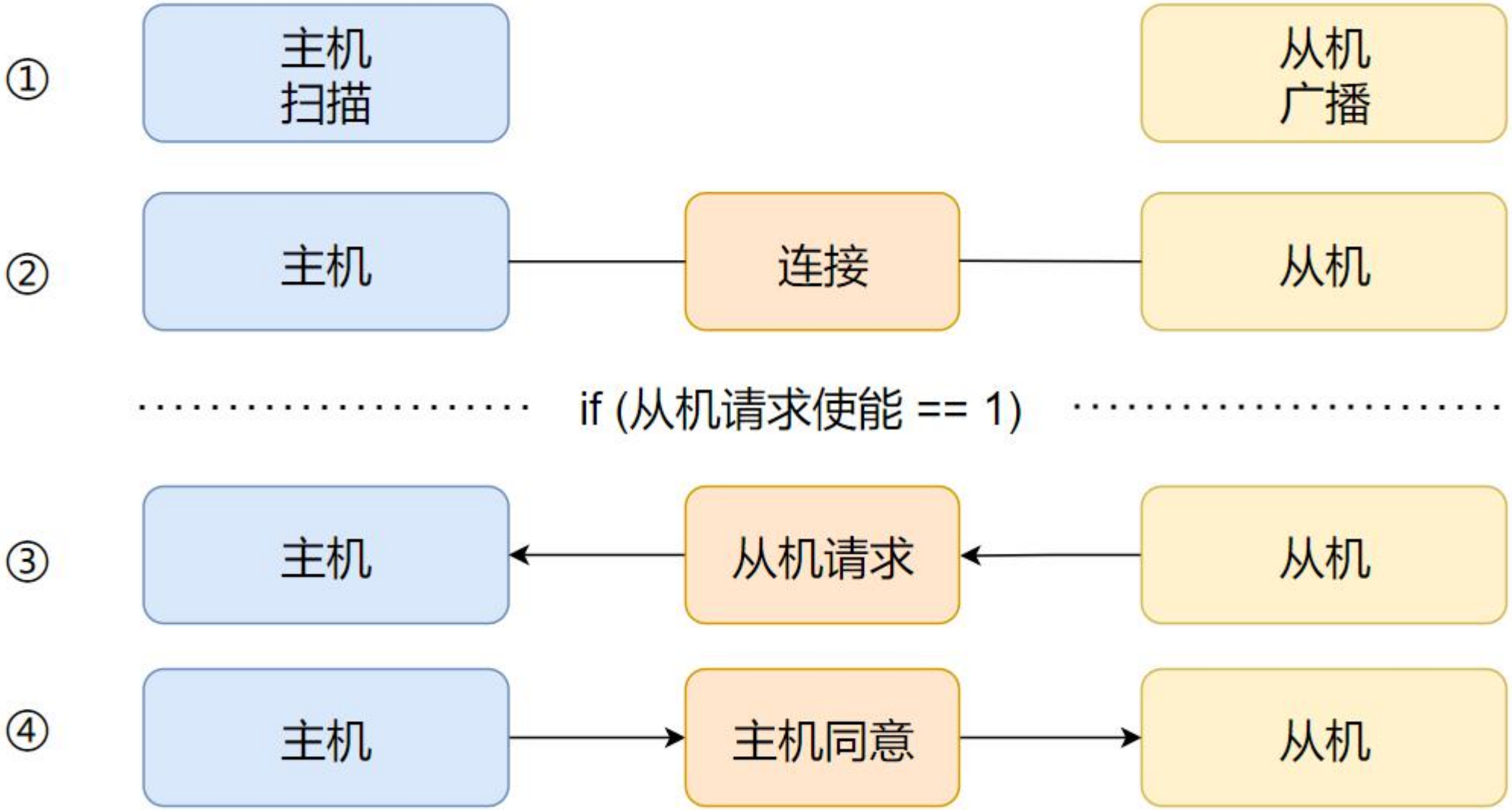
完整GATT从机连接间隔，latency，timeout

默认连接后连接间隔取最大值

```
#define ATT_SEND_CBUF_SIZE        (ATT_PACKET_NUMS_MAX * (ATT_PACKET_HEAD_SIZE

/* NOT_KEEP_RAM */
/* u8 gatt_ram_buffer[ATT_SEND_CBUF_SIZE + ATT_LOCAL_MTU_SIZE] __attribute__((

#define ADV_SCAN_MS(_ms)     ((_ms) * 8 / 5)
//搜索类型
#define SET_SCAN_TYPE        SCAN_ACTIVE
//搜索 周期大小
#define SET_SCAN_INTERVAL    ADV_SCAN_MS(24) // unit: 0.625ms
//搜索 窗口大小
#define SET_SCAN_WINDOW      ADV_SCAN_MS(8)  // unit: 0.625ms ,<= SET_SCAN_INTE

//连接周期
#define BASE_INTERVAL_MIN    (6)//最小的interval
#define SET_CONN_INTERVAL    (BASE_INTERVAL_MIN*4) //(unit:1.25ms)
//连接latency
#define SET_CONN_LATENCY     0  //(unit:conn_interval)
//连接超时
#define SET_CONN_TIMEOUT     100 //(unit:10ms)

//建立连接超时
#define SET_CREAT_CONN_TIMEOUT     0 //(unit:ms)

static u8 dg_pair_reconnect_search_profile = 0; /*配对回连是否搜索profile*/
//-------------------------------------------------------------------------
static scan_conn_cfg_t dg_central_scan_cfg;
static u16 dg_ble_central_write_handle;
static u16 dg_ble_central_read_handle;
static u16 dg_conn_handle, dg_conn_handle2;
static u8  cur_connect_pair_process;

//配对信息
#define   CLIENT_PAIR_BOND_ENABLE     CONFIG_BT_SM_SUPPORT_ENABLE
#define   PAIR_BOND_TAG               0x53
struct ctl_pair_info_t {
    u8 head_tag;
    u8 match_dev_id; //client_match_cfg_t 搜索表，设备的顺序0~
    u8 pair_flag;
    u8 peer_address_info[7];
    u16 conn_handle;
    u16 conn_interval;
    u16 conn_latency;
    u16 conn_timeout;
    u16 write_handle;
};
static u8 bt_connected = 0;
NORMAL  uc03_toy_flash  <pp_and_le/examples/dongle/ble_dg_central.c    c  utf
```

默认连接后连接间隔取最大值

四、SDK应用配置说明 - 蓝牙相关配置说明



谢谢观看

珠海研发项目组联系方式：

刘　杰　　电话：13928041563