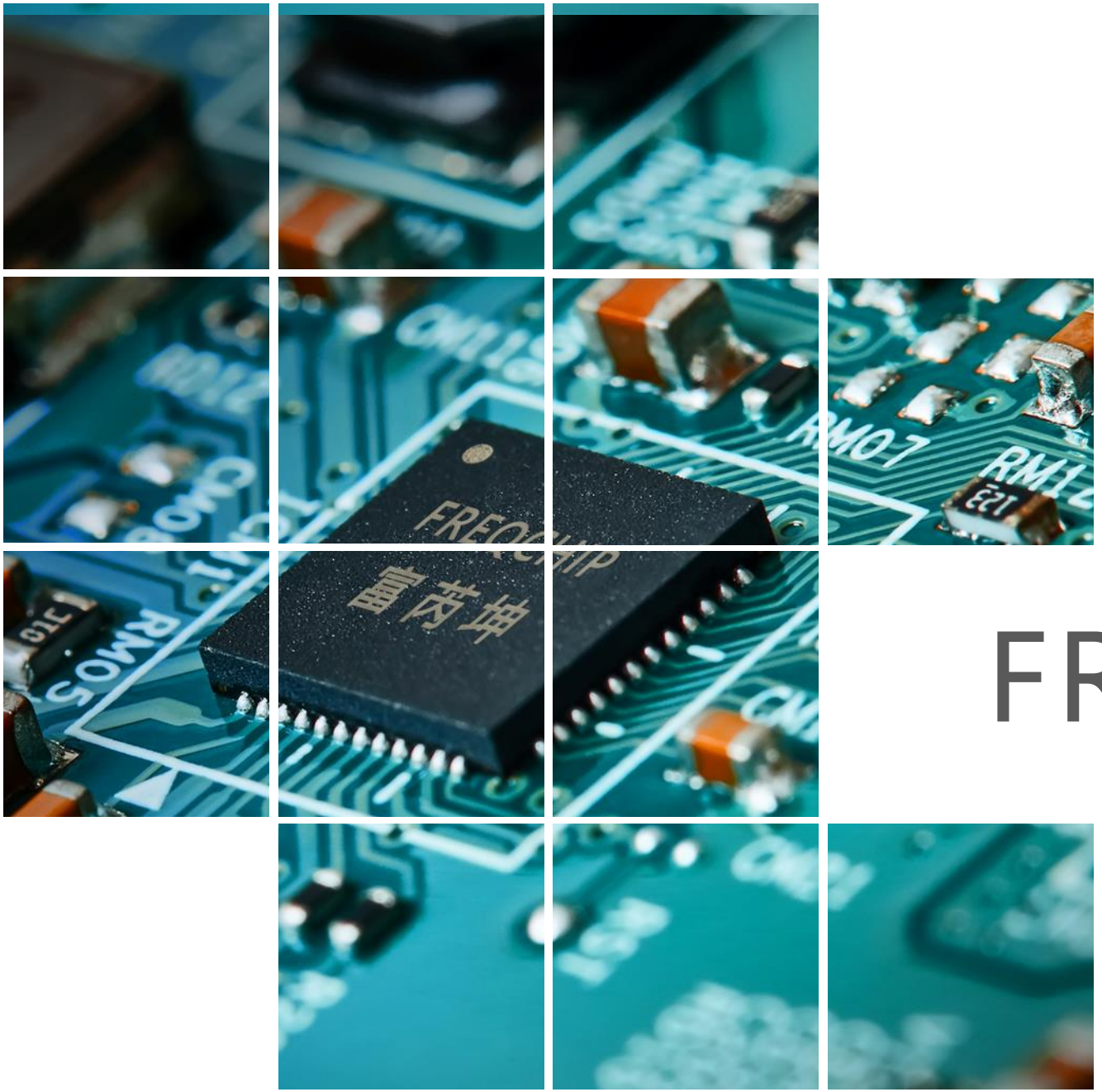


FREQCHIP 富芮坤

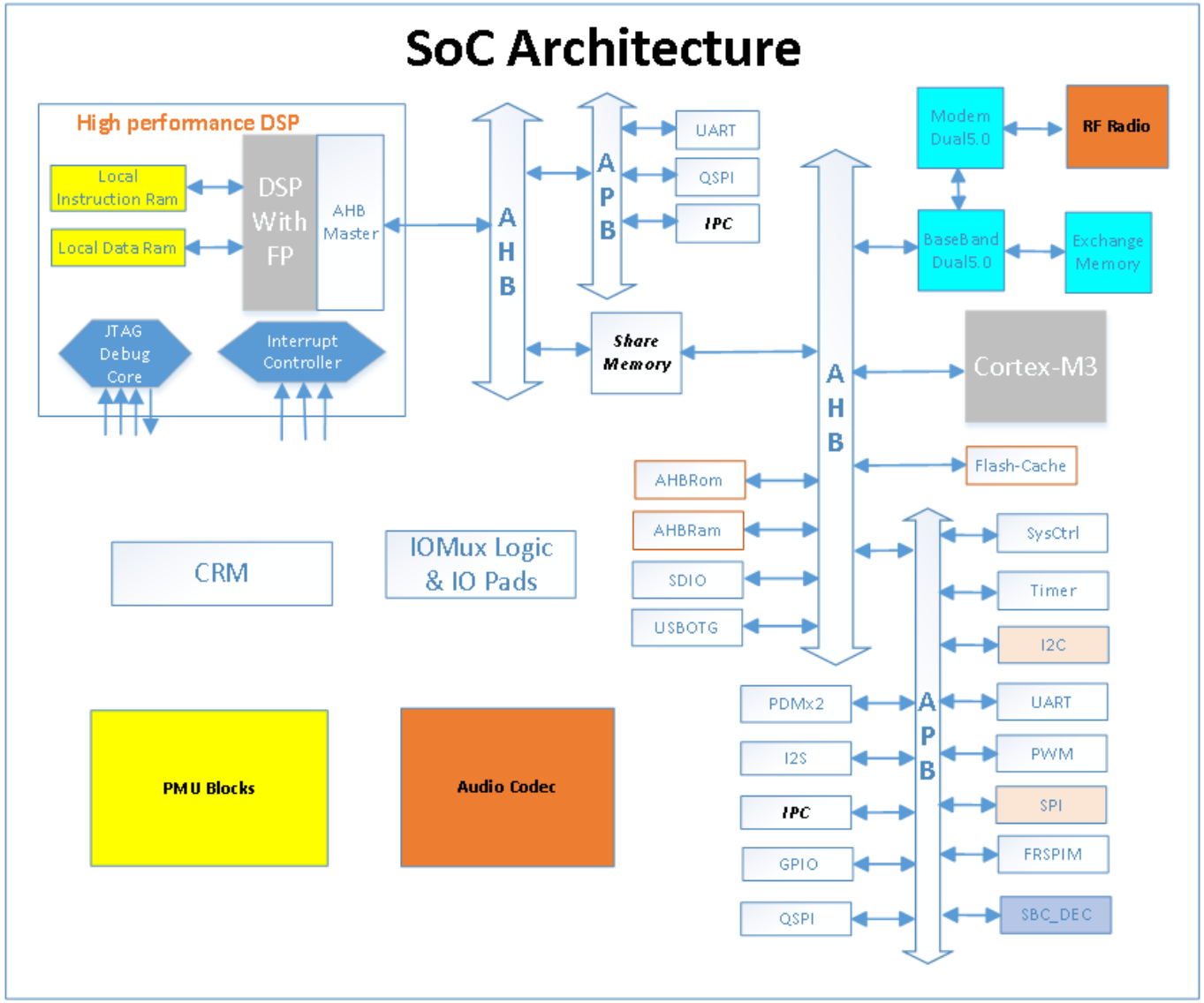
FR508x SDK介绍

FR508x硬件结构

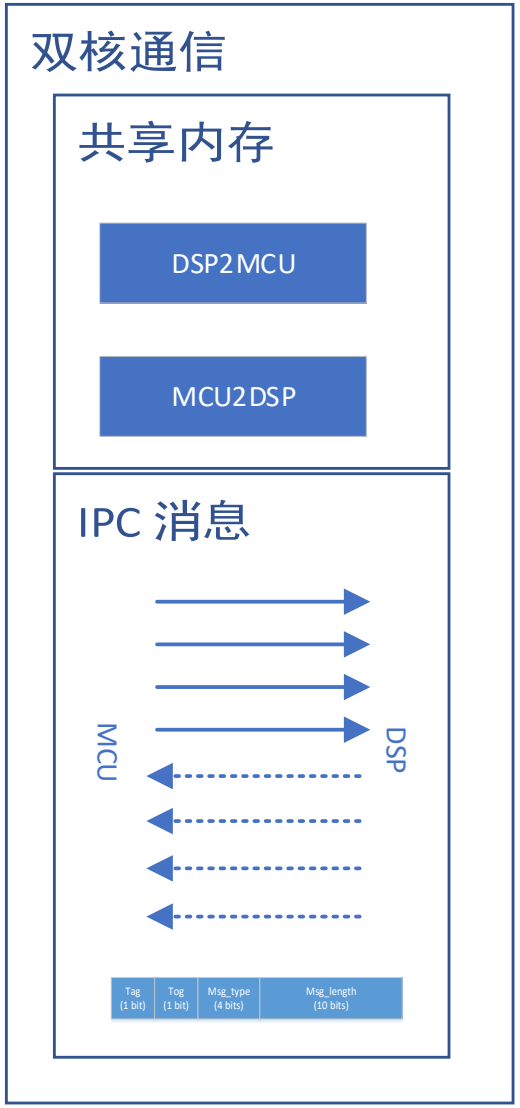
MCU:
- 48MHz主频
- 96KB RAM
- 512KB Flash
- UART
- Audio Codec
- Dual PDM
- SPI
- I2C
- USB OTG
- Timer
- PWM
- ……

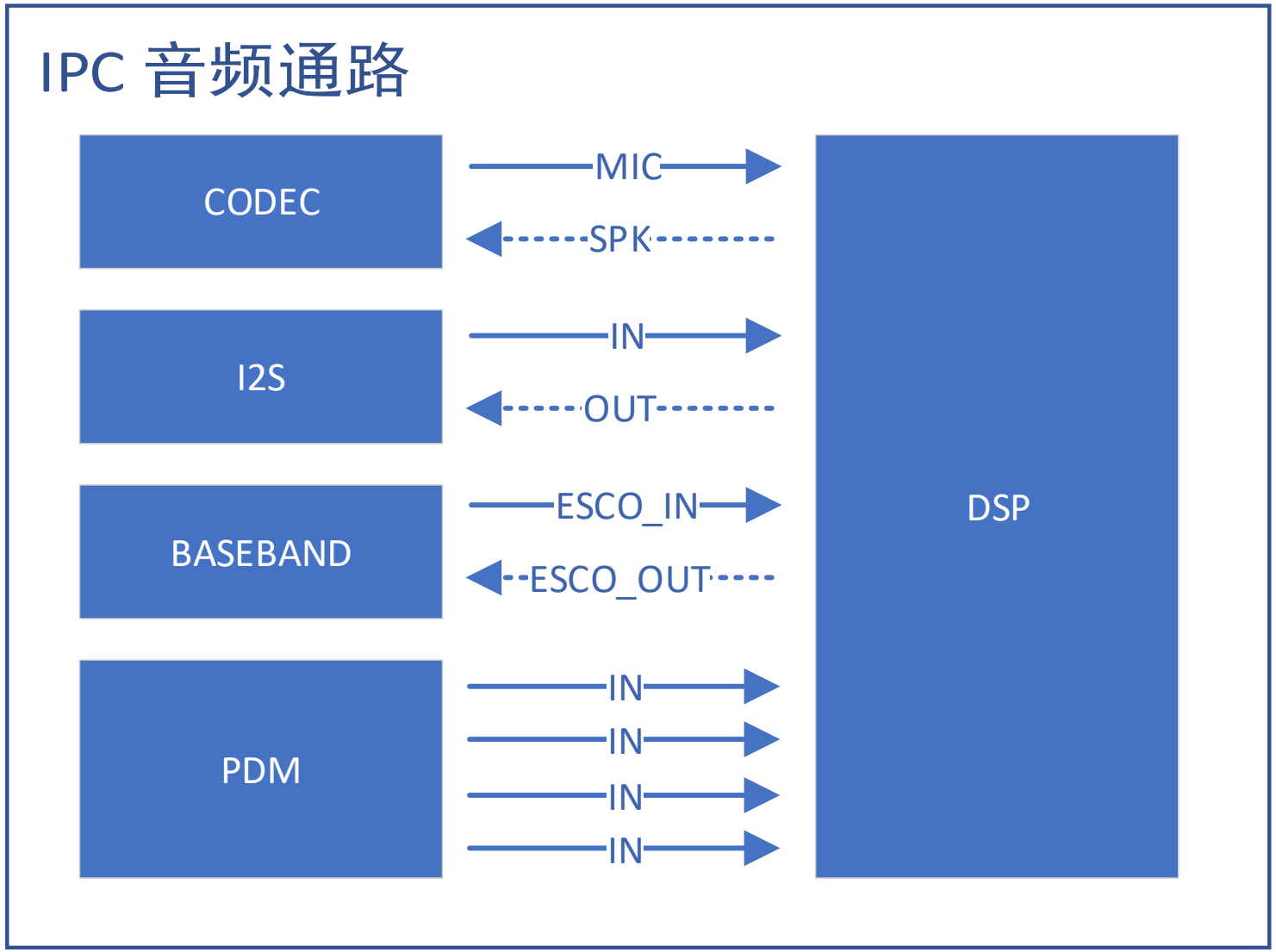
DSP:
- 156MHz主频
- 416KB SRAM
- UART
- QSPI
- 8路 GPIO

双核通信:
- 双向各4路IPC通道
- 双向各1KB共享内存
- 音频专用的IPC DMA通路



FR508x IPC

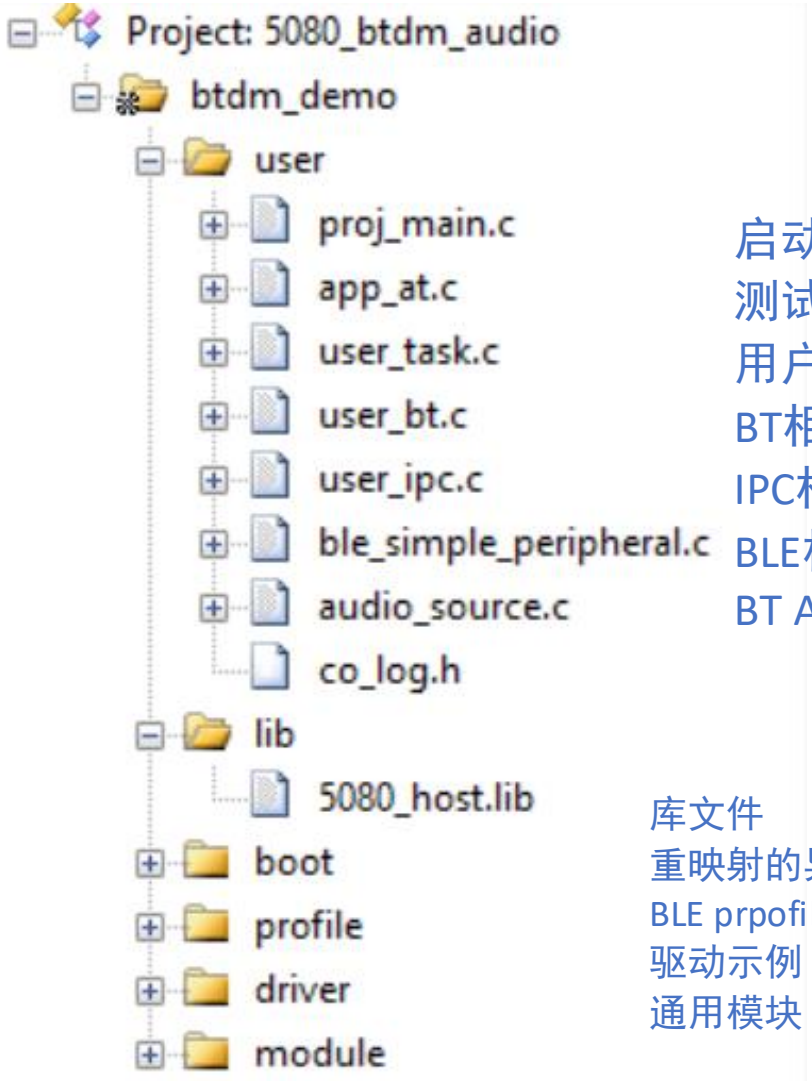

双核通信
共享内存
DSP2MCU
MCU2DSP
IPC 消息
MCU
DSP
Tag (1 bit)
Tog (1 bit)
Msg_type (4 bits)
Msg_length (10 bits)


IPC 音频通路
CODEC
MIC
SPK
I2S
IN
OUT
BASEBAND
ESCO_IN
ESCO_OUT
PDM
IN
IN
IN
IN
DSP

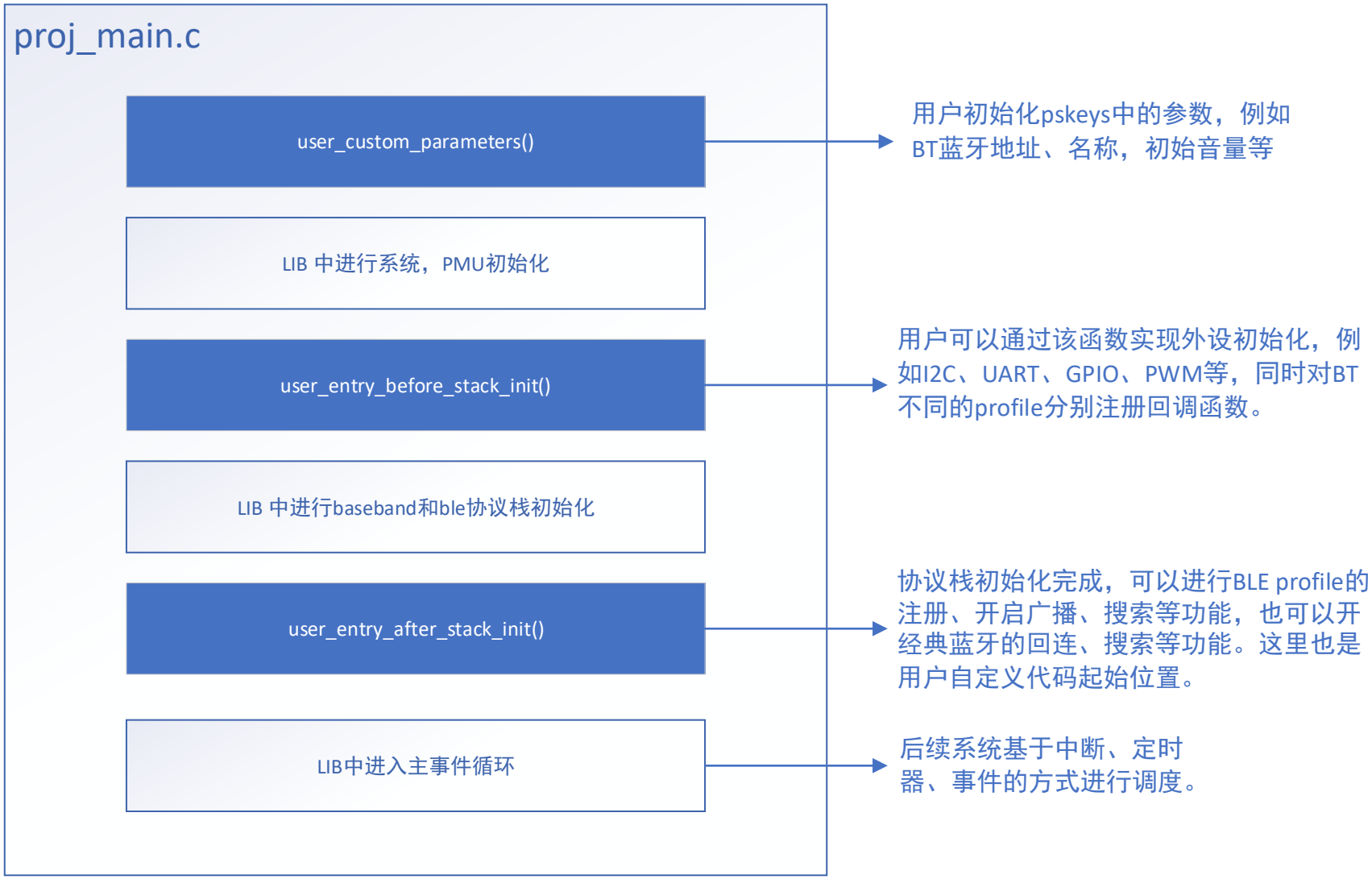
FR508x MCU SDK 结构

支持功能:

- BT通话
- BT Audio Sink
- BT Audio Source
- BLE
- 与DSP的双核通信
- 部分外设驱动

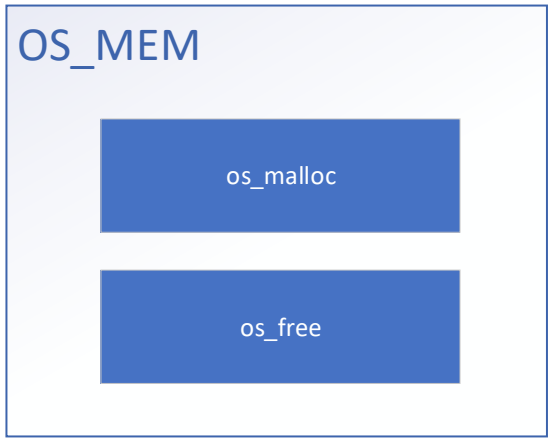


FR508x MCU启动流程



FR508x MCU测试用AT指令

FR508x MCU用户自定义任务及其他系统接口



注意事项：

1. 目前的SDK只能创建一个task
2. 协议栈的调度也是基于这个task
3. 对于一个事件的处理时间不能太长

注意事项：

1. 基于baseband时钟的一个软件timer
2. 定时精度在10ms
3. 对于一个timer事件的处理时间不能太长

FR508x ME回调和API

常用事件

BTEVENT_INQUIRY_RESULT：上报inqury的结果
BTEVENT_INQUIRY_COMPLETE：结束inquiry
BTEVENT_INQUIRY_CANCELED：取消inquiry
BTEVENT_LINK_CONNECT_IND：手机发起连接
BTEVENT_LINK_CONNECT_CNF：本机发起的链接建立好
BTEVENT_LINK_DISCONNECT：连接断开
BTEVENT_ACCESSIBLE_CHANGE：inquiry scan、page scan状态改变
BTEVENT_MODE_CHANGE：sniff和正常模式间的切换

常用API

bt_set_accessible_mode_nc：设置在未连接情况下BT的inquiry scan和page scan状态
bt_set_accessible_mode_c：设置在连接情况下BT的inquiry scan和page scan状态
bt_start_inquiry：开始搜索周围设备
bt_cancel_inquiry：取消搜索
bt_start_sniff：请求进入sniff
bt_stop_sniff：退出sniff
bt_switch_role：切换角色

FR508x HF 回调和API

常用事件

HF_EVENT_SERVICE_CONNECT_REQ：对端发来的连接请求
HF_EVENT_SERVICE_CONNECTED：与对端建立好HF连接
HF_EVENT_SERVICE_DISCONNECTED：HF连接断开
HF_EVENT_AUDIO_CONNECTED：与对端建立好SCO连接
HF_EVENT_AUDIO_DISCONNECTED：SCO连接断开
HF_EVENT_CALL_IND：通话状态
HF_EVENT_CALLSETUP_IND：通过建立过程状态指示
HF_EVENT_CALLER_ID_NOTIFY：来电号码
HF_EVENT_RING_IND：来电提示
HF_EVENT_COMMAND_COMPLETE：本地发出指令完成

常用API

hf_connect：发起HF连接
hf_disconnect：断开HF连接
hf_create_audio_link：建立SCO连接
hf_disconnect_audio_link：断开SCO连接
hf_answer_call：接听电话
hf_dial_number：拨号
hf_redial：重播
hf_hang_up：挂断
hf_generate_dtmf：发出DTMF
hf_disable_nrec：关闭手机NREC

FR508x A2DP 回调和API

常用事件

A2DP_EVENT_STREAM_OPEN：与对端的A2DP建立好连接
A2DP_EVENT_STREAM_CLOSED：与对端的A2DP连接断开
A2DP_EVENT_STREAM_STARTED：开始播放音频数据流
A2DP_EVENT_STREAM_SUSPENDED：停止播放音频数据流
A2DP_EVENT_STREAM_SBC_PACKET_SENT：A2DP SOURCE模式下成功发送一个A2DP数据包给到对端

常用API

a2dp_open_stream：建立与对端的A2DP连接
a2dp_start_stream：请求开始A2DP数据流传输
a2dp_suspend_stream：停止A2DP数据流传输
a2dp_stream_send_sbc_packet：A2DP SOURCE模式下发送A2DP数据包给对端

FR508x AVRCP 回调和API

常用事件

AVRCP_EVENT_CONNECT：与对端的AVRCP建立好连接
AVRCP_EVENT_DISCONNECT：与对端的AVRCP连接断开
AVRCP_EVENT_PANEL_CNF：发送的按键消息被对端接收

常用API

avrcp_set_panel_key：发送按键消息给手机，比如上一曲、下一曲、快进、快退、暂停等。

FR508x SPP 回调和API

常用事件

SPP_EVENT_REMDEV_CONNECTED：与对端的SPP建立好连接
SPP_EVENT_REMDEV_DISCONNECTED：与对端的SPP连接断开
SPP_EVENT_DATA_IND：收到对端传来的数据
SPP_EVENT_SEND_COMPLETE：本地发送的数据到了对端

常用API

spp_connect：向指定地址的设备发起SPP连接
spp_disconnect：断开与对端的SPP连接
spp_send_data：发送数据给对端

FR508x BLE相关操作

BLE 初始化

- 可选择的初始化
- 自定义profile，并添加注册
- 注册GAP回调函数

BLE 常用API

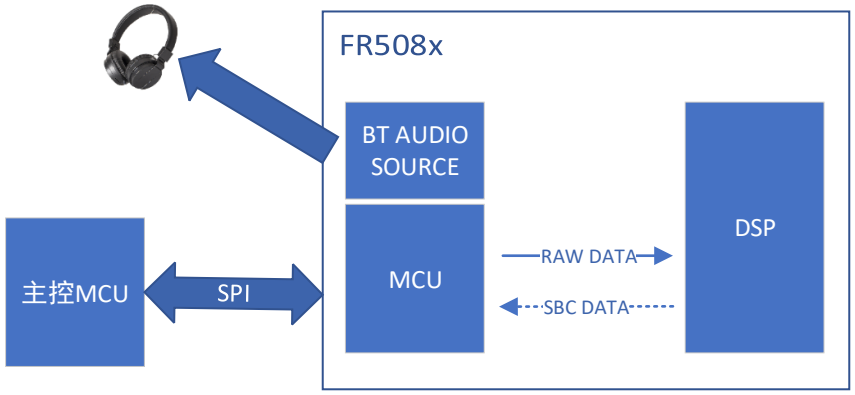
gap_set_advertising_param
gap_set_advertising_data
gap_set_advertising_rsp_data
gap_start_advertising
gap_stop_advertising
gap_start_conn
gap_disconnect_req
...

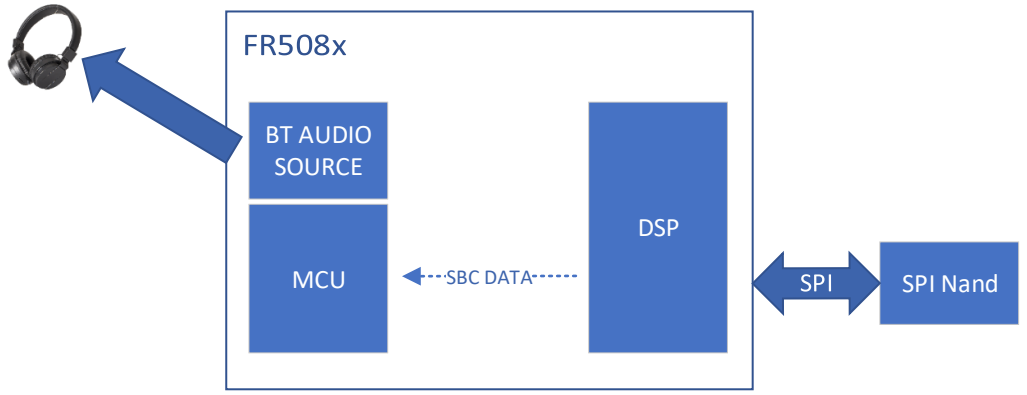
```
gap_security_param_init(&param);

gap_address_set(&addr);

gap_set_dev_name(local_name, sizeof(local_name));
```

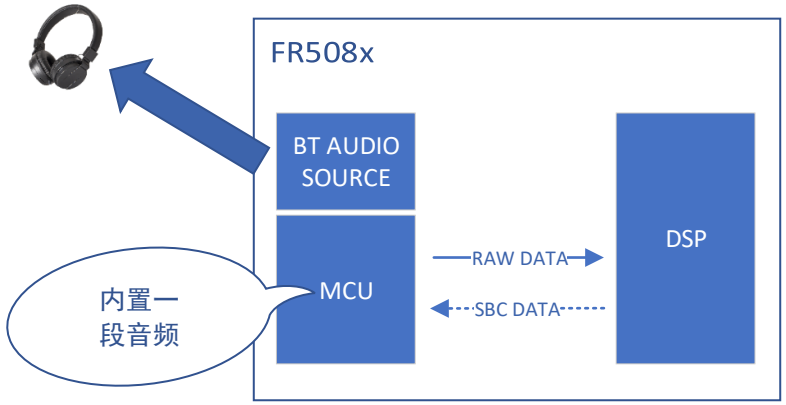
```
const gatt_attribute_t simple_profile_att_table[SP_IDX_NB] =
{

static void sp_gatt_read_cb(uint8_t *p_read, uint16_t *len, uint16_t att_idx)
{

static void sp_gatt_write_cb(uint8_t *write_buf, uint16_t len, uint16_t att_idx)
{

static uint16_t sp_gatt_msg_handler(gatt_msg_t *p_msg)
{

void sp_gatt_add_service(void)
{
    simple_profile_svc.p_att_tb = simple_profile_att_table;
    simple_profile_svc.att_nb = SP_IDX_NB;
    simple_profile_svc.gatt_msg_handler = sp_gatt_msg_handler;

    sp_svc_id = gatt_add_service(&simple_profile_svc);
}
```

```
 */
void app_gap_evt_cb(gap_event_t *p_event)
{

    gap_set_cb_func(app_gap_evt_cb);
```

FR508x Audio Source

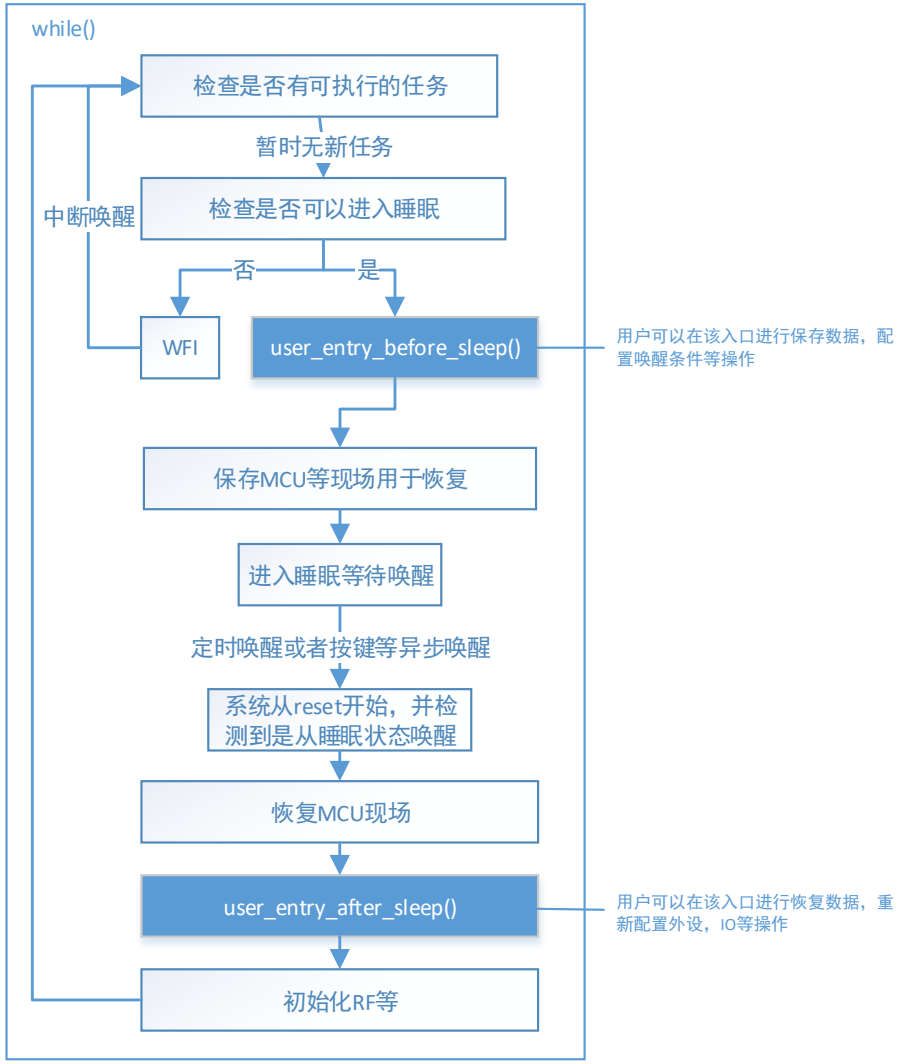






1. 与耳机建立连接
2. 启动DSP
3. 开始A2DP
4. DSP请求原始数据
5. MCU向DSP请求SBC数据
6. MCU将收到的SBC数据传输给耳机
7. 4~6循环进行

FR508x 低功耗机制



入口函数

1.user_entry_before_sleep：该函数在进入睡眠前被调用，用户可在里面实现控制GPIO的状态保持（针对GPIO在系统工作和睡眠状态下的控制参见外设驱动章节），配置睡眠唤醒条件等行为。

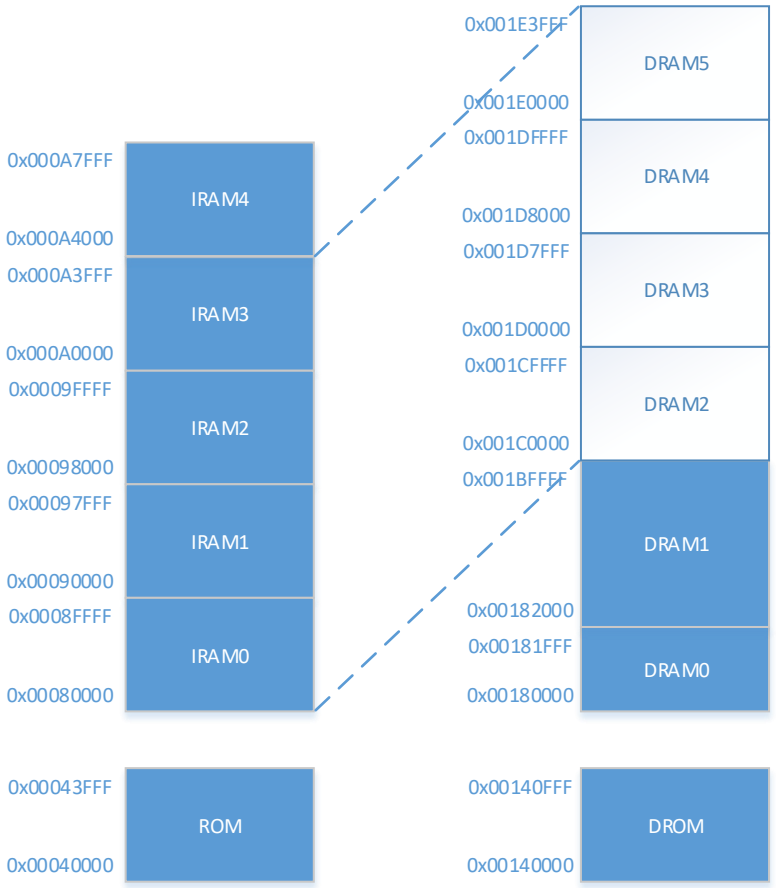
2.user_entry_after_sleep：在系统唤醒后，用户可以在该函数中重新进行外设的初始化（进入睡眠后外设的状态因为掉电都会丢失）等操作。

唤醒条件

1.同步唤醒：同步唤醒来自一个硬件timer，这个timer的设置由协议栈中代码完成，主要取决于BT pagescan和inquiry sacn间隔、BT sniff间隔、 BLE广播间隔、BLE连接间隔等参数，在应用层代码中无需关注。

2.异步唤醒：异步主要来自于PMU（电源管理单元）的中断信号，PMU的中断源有：充电器插入拔出、RTC、GPIO状态监测模块等，这些中断源可以在系统初始化时进行设置。

FR508x DSP存储结构



1. 总共可用内存为416KB
2. 默认iram为160KB，dram为256KB
3. Iram可以将部分内存划分到dram
4. 内存的划分由M3来进行控制

FR508x DSP的控制

DSP的上下电

1. pmu_power_on_DSP：给DSP上电。
2. pmu_power_off_DSP: DSP断电。

```
enum system_dsp_clk_src_sel_t {
    SYSTEM_DSP_SRC_OSC_48M,
    SYSTEM_DSP_SRC_OSC_48M_DIV,
    SYSTEM_DSP_SRC_208M,
    SYSTEM_DSP_SRC_312M,
};
```

DSP的运行时钟配置

1. system_set_dsp_clk：设置DSP的运行频率。
2. QSPI reference clock的配置。

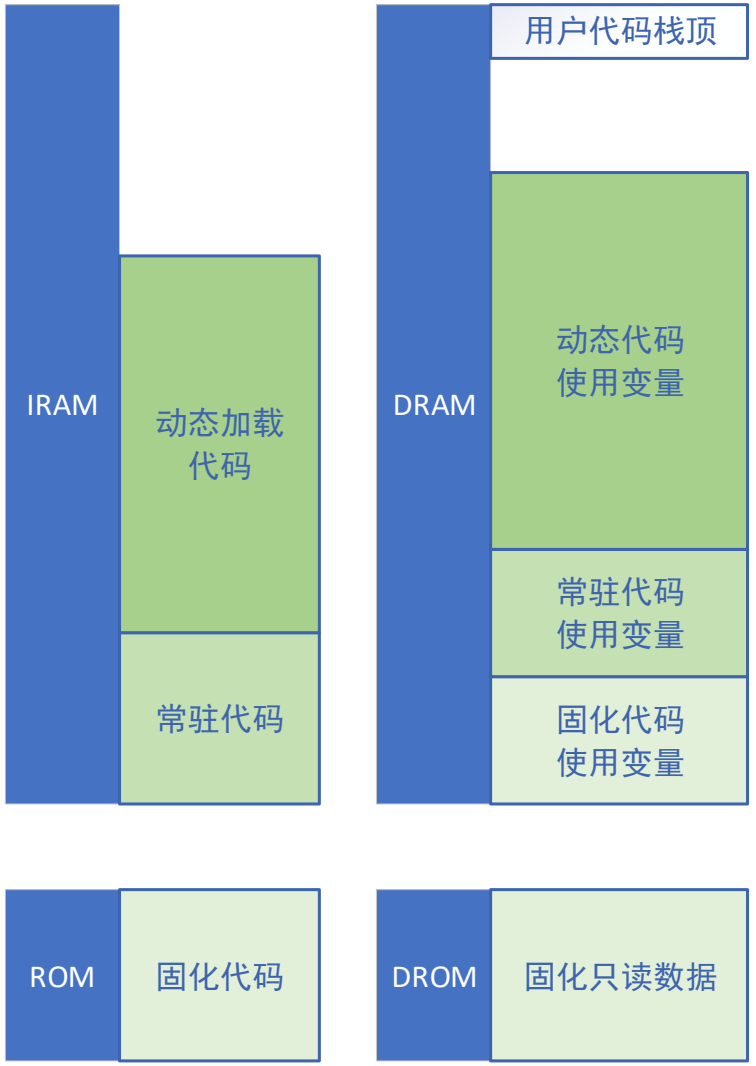
```
struct syscntl_clock_ctrl_t{
    uint32_t clk_bb_div:3;
    ///000---48M, other--- 48M/(clk_bb_div+1)
    uint32_t clk_lp_sel:1;
    ///0---rc 62.5K, 1---32K from osc24M divid
    uint32_t clk_gpio_sel:1;
    ///0---cm3 master clock,1---deep sleep clo
    uint32_t dsp_clk_sel:2;
    ///00---osc 48M,  01---osc 48M/(2*(dsp_mas
    ///10---spll 208M/(2*(dsp_mas_clk_div + 1)
    uint32_t cm3_qspi_ref_clk_sel:1;
    ///0---48M clock,1---52M clock
    uint32_t out_clk_div:8;
    ///output clock divider
    uint32_t dsp_mas_clk_div:8;
    ///dsp master clock divider
    uint32_t clk_out_sel:1;
    ///0---osc/out_div, 1---32K from osc24M di
    uint32_t adc_clk_sel:1;
    ///0---high speed clok from clock divider,
    uint32_t adc_clk_div:4;
    ///48M/clkdiv
    uint32_t resv:1;
    uint32_t bb_phase_sel:1;
    ///0---normal,1---48M clock negedge
};
```

DSP的复位与复位向量地址

1. system_set_dsp_vector：设置DSP的复位向量地址
2. system_reset_dsp_set：将DSP设置为复位状态
3. system_reset_dsp_release：将DSP从复位状态释放
4. system_reset_dsp：复位DSP

1. ROM空间代码的复位向量地址为0x40000
2. iRAM空间的代码复位向量地址为0x80000

FR508x DSP代码结构（非XIP结构）

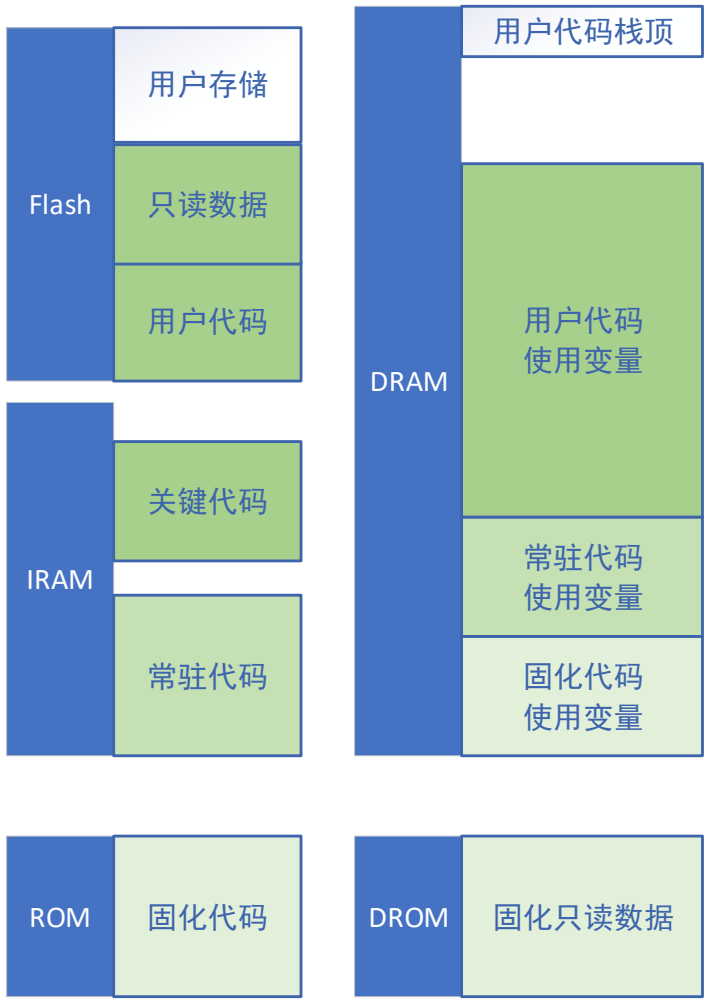


1. 代码分为basic和user_code两部分
2. Basic代码为基础代码，主要功能是bootloader，由ROM code负责加载
3. User_code为用户代码，属于动态按需加载的部分。由basic代码负责加载

优点：所有代码运行在iram，运行速度快。
缺点：用户代码比较大的时候加载速度比较慢。

使用场景：功耗要求比较小（白电的语音识别），时间相应比较慢（做协处理器时的音频编解码）等。

FR508x DSP代码结构（XIP结构）

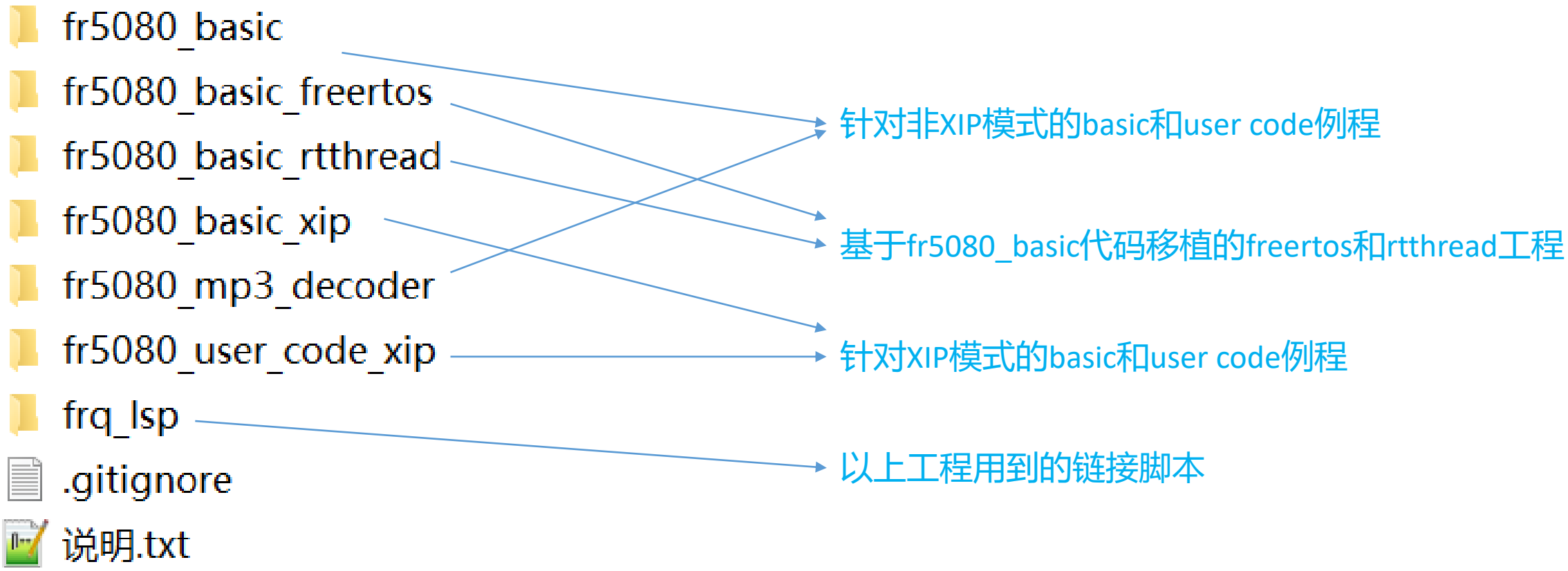


1. 代码分为basic和user_code两部分
2. Basic代码为基础代码，主要功能是bootloader，由ROM code负责加载
3. User_code为用户代码，运行在DSP的外挂flash上，运行在XIP模式，无需额外加载。

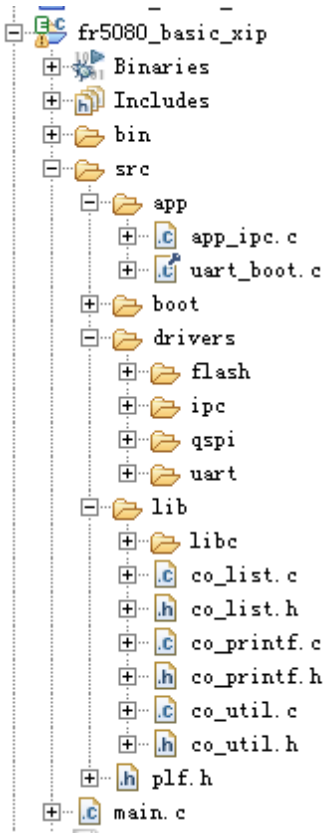
优点：只需额外加载basic代码，启动速度快。
缺点：用户代码运行在XIP模式，运行速度相对于iram中运行速度慢一些。

使用场景： 单芯片穿戴设备上，有推屏的需求等场景。

FR508x DSP SDK目录结构

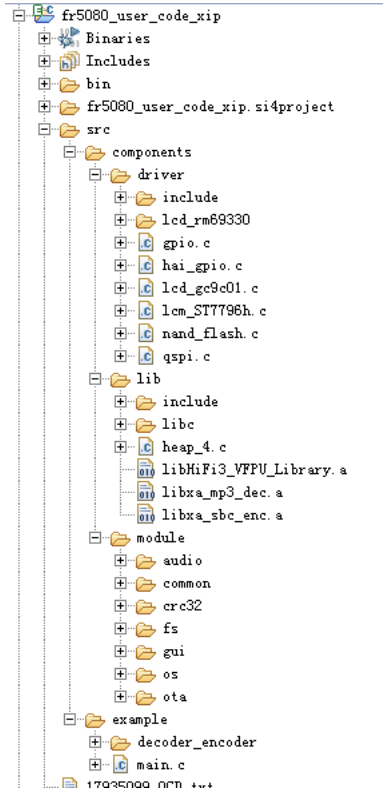


FR508x DSP SDK提供的API（以XIP模式为例）



Basic代码：

1. 主要功能为bootloader
2. 提供ipc、uart等驱动接口
3. 提供了一些库函数



User code 代码：

1. 提供了部分GPIO、LCD驱动程序
2. 提供了DSP的函数库
3. 提供了内存分配函数
4. 提供了GUI、CRC等常用接口
5. 提供了MP3解码和SBC编码的例程

FR508x DSP SDK提供的API（以XIP模式为例）

1. 创建消息

void task_msg_insert(void *task_msg);
void *task_msg_alloc(uint32_t id, uint32_t msg_size);

2. 软件timer的使用

void os_timer_init(struct os_timer_t *timer, os_timer_func_t callback, void *parg);
void os_timer_destroy(struct os_timer_t *ptimer);
void os_timer_start(struct os_timer_t *timer, uint32_t period_ms, enum os_timer_type_t type);
void os_timer_stop(struct os_timer_t * timer);

3. 内存管理

void *pvPortMalloc(size_t xWantedSize)
void vPortFree(void *pv)

4. 其它

void app_ipc_rx_set_user_handler(void *arg);
……

FR508x IPC通信示例

Tag (1 bit)	Tog (1 bit)	Msg_type (4 bits)	Msg_length (10 bits)

1. 这是一个硬件寄存器
2. MCU和DSP端各有4组接收用寄存器和发送用寄存器
3. TAG字段在发送寄存器中写1为通知对方消息准备完成
4. TAG字段在接收寄存器中写0为通知对方消息处理完成
5. TOG:表示当前数传的内容在payload的情况
6. 剩余字段为软件分配，目前定义成上面的结构
7. 在SDK中已经提供好接口

```
enum ipc_user_msg_type_t {
    /*IPC_MSG_LOAD_CODE = 0,
    IPC_MSG_LOAD_CODE_DONE = 1,
    IPC_MSG_EXEC_USER_CODE = 2,
    IPC_MSG_DSP_READY = 10,*/
    IPC_MSG_RAW_FRAME = 3,
    IPC_MSG_DECODED_PCM_FRAME = 4,
    IPC_MSG_ENCODED_SBC_FRAME = 5,
    IPC_MSG_WITHOUT_PAYLOAD = 6,
    IPC_MSG_RAW_BUFFER_SPACE = 7,
};
```

```
enum ipc_user_sub_msg_type_t {
    IPC_SUB_MSG_NEED_MORE_SBC,
    IPC_SUB_MSG_DECODER_START,
    IPC_SUB_MSG_REINIT_DECODER,
    IPC_SUB_MSG_NREC_START,
    IPC_SUB_MSG_NREC_STOP,
    IPC_SUB_MSG_DECODER_STOP,
};
```

FR508x IPC通信示例

void ipc_msg_send(enum ipc_msg_type_t msg, uint16_t sub_msg, ipc_tx_callback callback)

void ipc_msg_with_payload_send(enum ipc_msg_type_t msg, void *header, uint16_t header_length,
uint8_t *payload, uint16_t payload_length, ipc_tx_callback callback)

1. MCU发送数据给DSP

```
void send_raw_data_to_dsp(uint8_t *data_buf, uint32_t data_len)
{
    CPU_SR cpu_sr;

    ipc_msg_with_payload_send((enum ipc_msg_type_t)IPC_MSG_RAW_FRAME, NULL, 0, data_buf, data_len, send_raw_data_to_dsp_done);
```

2. DSP端收到消息并处理

```
static void ipc_rx_user_handler(struct ipc_msg_t *msg, uint8_t chn)
{
    uint8_t channel;

    switch(msg->format) {
        case IPC_MSG_RAW_FRAME:
            mp3_decoder_recv_frame(ipc_get_buffer_offset(IPC_DIR_MCU2DSP, chn), msg->length);
            break;
        case IPC_MSG_WITHOUT_PAYLOAD:
            switch(msg->length) {
                case IPC_SUB_MSG_DECODER_START:
                    mp3_decoder_init();
                    break;
                case IPC_SUB_MSG_NEED_MORE_SBC:
                    sbc_encoder_recv_frame_req();
                    break;
```

3. MCU发送无payload消息给DSP

```
ipc_msg_send((enum ipc_msg_type_t)IPC_MSG_WITHOUT_PAYLOAD, IPC_SUB_MSG_NEED_MORE_SBC, 0);
```

4. DSP收到MCU的请求做出响应

```
void sbc_encoder_recv_frame_req(void)
{
    struct task_msg_t *msg;
    uint8_t left_frame_to_send = SBC_A2DP_SUB_FRAME_COUNT;
    uint8_t channel;
    UWORD32 length = 0;
    pUWORD8 buffer;
    pUWORD16 single_frame_len;

    printf("sbc_req before: %d.\r\n", sbc_encoder_encoded_frame_count);

    channel = ipc_alloc_channel(SBC_A2DP_FRAME_LENGTH);
    if(channel != 0xff) {
        buffer= ipc_get_buffer_offset(IPC_DIR_DSP2MCU, channel);
        single_frame_len =(pUWORD16)buffer;
        buffer += 2;
        length += 2;
        while((left_frame_to_send) && (sbc_encoder_encoded_frame_count)) {
            struct sbc_encoder_encoded_frame_t * frame;
            GLOBAL_INT_DISABLE();
```

FR508x DSP开发注意事项

1. 合理分配iram和dram
2. DSP的上下电、时钟设置都是由MCU来控制
3. XIP模式下注意Flash的四线使能，对于不同的Flash型号采用的方式有所不同
4. XIP模式下运行速度有要求的代码需要链接到iram里面

联系我们

上海 张江高科技园区碧波路912弄华依创新园8号楼5楼

青岛 青岛市崂山区科苑纬一路1号国际创新园D2座16层

深圳 宝安区西乡街道共和工业路35号红盒子联合办公213室

www.freqchip.com

sales@freqchip.com

021-50270080

微信公众号