Rockchip啸叫音频算法调试说明文档

文件标识：RK-KF-SF-980

发布版本：V1.0.0

日期：2024-07-18

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文提供进阶版本的啸叫抑制调试文档，工程师可以参照相关内容进行啸叫抑制算法的调试。基础版本在RKStudio中有集成，用户可根据需求选择算法版本。

产品版本

芯片名称	内核版本
全系列	通用

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	李茂发	2024-07-18	初始版本

目录

概述

音频算法特性

啸叫抑制算法，主要包括主动啸叫抑制、被动啸叫抑制模块，主动啸叫抑制和被动啸叫抑制模块可级联使用，即先主动啸叫抑制，再被动啸叫抑制，也可以分开使用。

音频算法说明

啸叫抑制主要完成啸叫抑制功能。

音频算法模块说明

完整的算法包括的流程模块如图1所示，包括如下部分：

主动啸叫抑制模块

主动啸叫模块包括AF模块、PF模块和AES模块。

- AF模块：时域反馈抑制算法，根据回采信号消除麦克风的反馈信号。需要主要该模块容易造成mic端语音过消，需要开启NLP模块进行mic端语音去相关，并且滤波器长度尽量小于mic端传输到播放的时间，以确保不过消。
- PF模块：频域反馈抑制算法，需要注意该方法会带来额外的1帧算法延迟，其余注意用法和AF模块一致。
- AES模块：反馈抑制的后处理频域算法，需要注意该方法会带来额外的1帧算法延迟，AES必须在AF或者PF开启后使用。

被动啸叫模块

被动啸叫模块包括NOTCH模块、SHIFT模块和VC模块。

- NOTCH模块：属于时域陷波检测和抑制算法，该模块不需要采用回采，无特殊适配要求。
- SHIFT模块：移频模块，进行固定频率的频移，一般控制在3Hz左右。
- VC模块：移频模块，按比例进行频率的移动，越到高频偏移的频率越大。

辅助模块

辅助模块包括NLP模块、HPF模块和LIMIT模块。

- NLP模块：语音去相关模块，需要注意，该模块会带来一定的语音失真。
- HPF模块：高通滤波器，用于滤除部分低频。
- LIMIT模块：时域限幅器，用于过滤底噪以及防止预增益以及其他操作引起截幅反转现象。

音频算法流程

啸叫抑制算法的语音通路如图1所示。

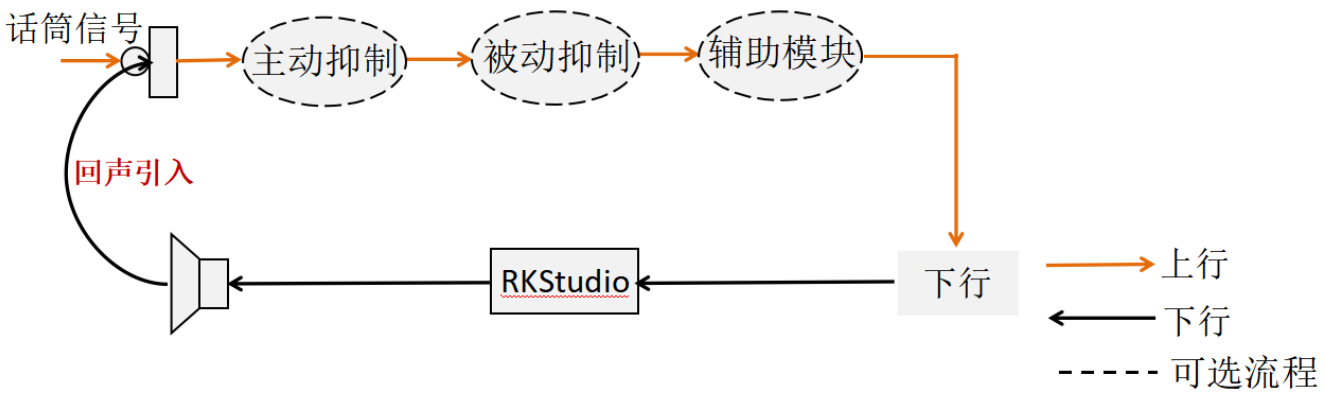

图 1 啸叫抑制流程系统框图

1. 阵列采集阵列信息后，先通过主动抑制模块消除回声（可选）；
2. 消除部分后的回声数据，通过被动抑制在进行啸叫抑制（可选）；
3. 最后送入辅助模块进行数据的处理（可选）。

主动抑制算法流程

主动抑制算法的语音通路如图2所示。

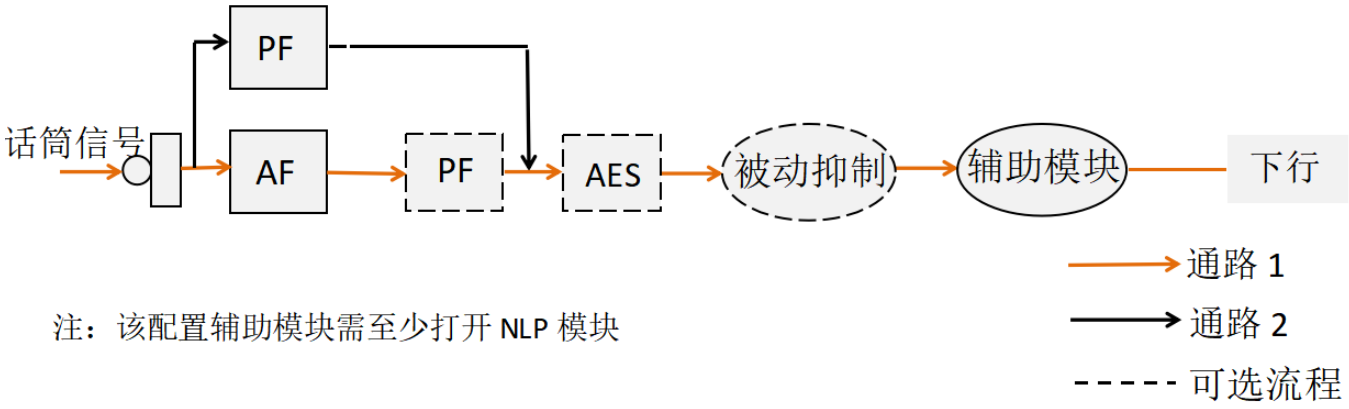

图 2 主动抑制流程系统框图

1. 阵列采集阵列信息后，可选通路1即通过AF进行线性回声消除，AF后可通过PF或者AES进一步消除回声，注意PF或者AES会带来1帧延迟（该延迟共享，即单开PF、AES和两者全开，均增加一帧延迟）；
2. 阵列采集阵列信息后，可选通路2即通过PF进行线性回声消除，PF后可通过AES进一步消除回声。该方案只推荐用于消除音乐声，并且会带来固定一帧延迟；
3. 主动抑制后可送入被动抑制模块（可选）；
4. 主动抑制后可送入辅助模块，辅助模块必开，且需使能NLP模块。

被动抑制算法流程

被动抑制算法的语音通路如图3所示。

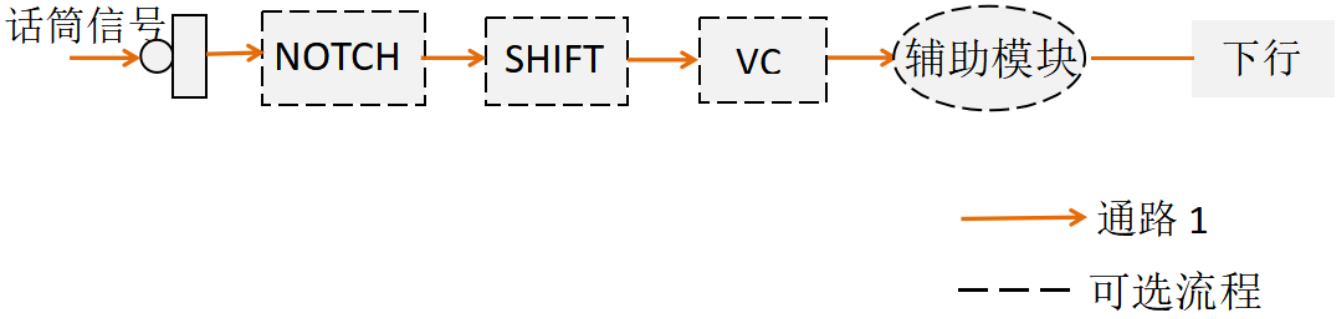

图 3 被动抑制流程系统框图

1. 阵列采集阵列信息后，可依次通过NOTCH、SHIFT和VC模块（三个模块均可选）；
2. 主动抑制后可送入辅助模块（可选）。

辅助模块算法流程

辅助模块算法的语音通路如图4所示。

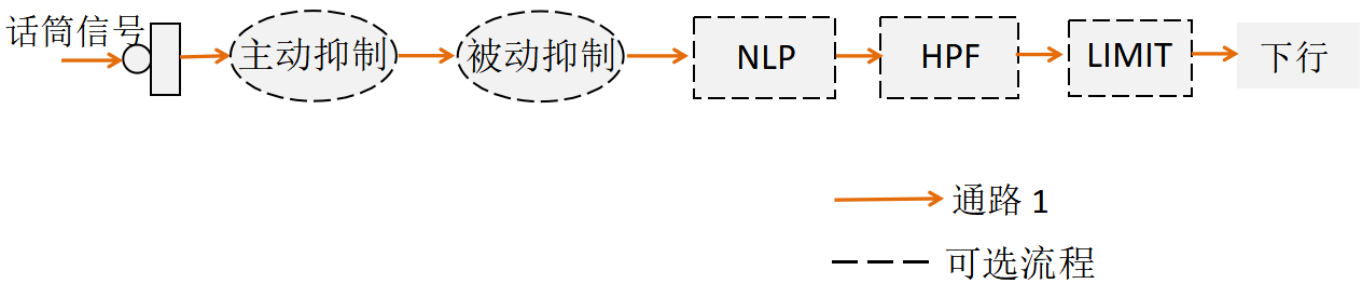

图 4 辅助模块流程系统框图

1. 啸叫处理后的数据，可依次通过NLP、HPF和LIMIT模块（三个模块均可选）；

音频算法接口调用说明

该库的调用过程如下：

1. 设置参数：

该步骤用于对啸叫抑制算法开放的音频参数进行设置，参数设置需要通过rkaudio_howling_interface.h文件调整也可以通过config_howling.json文件配置，将该文件路径名作为初始化的输入参数。

2. 初始化：

该步骤用于算法开启后，逐帧处理声音信号。初始化接口函数名为rkaudio_howling_init，具体调用如下所示：

```
{

  st_ptr = rkaudio_howling_init(mSampleRate, mBitPerSample, num_src_channel,
  num_ref_channel, param);

}
```

其中，mSampleRate采样率支持8kHz、16kHz、24kHz、32kHz、44.1kHz、48kHz，mBitPerSample音频位深为16bit，num_src_channel为mic通道数，num_ref_channel为回采通道数，param为输入参数，该参数通过rkaudio_howling_interface.h配置将在后面进行详细说明。

初始化完成算法内部参数初始化和相关内存申请，若初始化成功则返回st_ptr结构体，如果初始化失败，结构体将为NULL。

3. 帧处理：

该步骤即是实时处理语音信号，其单位为帧，当固定每帧2.5ms、采样率为48k采样时，每帧包含128个样点。帧处理包括接口函数为rkaudio_howling_short，具体调用如下所示：

```
{

  out_size = rkaudio_howling_short(st_ptr, in, out, in_size);

}
```

其中，st_ptr为初始化后生成的结构体，in为输入音频，单位为short，out为输出音频，单位为short，in_size为一帧读入的short数，即`帧长 * num_channel`，out_size为一帧输出的byte数，即`帧长 * num_src_channel`。

4. 销毁释放：

该步骤是在通话结束后，用于释放算法内存变量，接口函数为rkaudio_howling_destory(st_ptr)rkaudio_howling_param_destory(param)。

注：上述示例仅针对于.h配置，json配置相关请参考调用示例Main.cpp

音频算法参数调试说明

啸叫算法通过rkaudio_howling_interface.h文件进行参数配置。

全局模块参数设置

下面是对输入音频各种通道数的基础设置。

```
#define REF_CHANNEL     2   /* 设置回采通道数，该处设置2回采通道 */
#define REF_POSITION    1   /* 回采位置，0为回采通道位于mic数据最前面，1则在最后面 */
```

主动抑制模块参数设置

其中各子模块开关主要通过**model_en**控制，主要包括：*RKAUDIO_EN_AF*、*RKAUDIO_EN_PF*、*RKAUDIO_EN_AES*。具体可看下面代码注释。

AF子模块参数设置

AF模块通过rkaudio_af_param_init()函数配置，主要参数如下：

```
param->filter_len = 3;            /* AF滤波器长度，实际可覆盖回声长度为filter_len * 帧长 */
param->delay_len = 0;             /* 回采延迟长度，即对回采进行延迟，实际延迟长度为delay_len * 帧长 */
param->af_ref_threshold = 10;     /* 回采vad阈值，幅值高于该值判定为有效回采，低于该值算法不做处理 */
param->af_mic_threshold = 10;     /* 话筒vad阈值，幅值高于该值判定为有效录音，低于该值算法不做处理 */
param->start_freq = 200;          /* AF处理起始频率，该值必须大于0 */
param->end_freq = 48000;          /* AF处理结束频率，该值必须小于采样率 */
param->adapt_rate = 0.15;         /* 滤波器收敛速度，过快容易导致语音过消和滤波器分散，过慢则消回声收敛慢 */
param->jump_len = 0;              /* 跳帧进行滤波器更新，节省算力 */
param->auto_adapted = true;       /* 是否开启滤波器步长自适应，开启后可加快回声收敛速度 */
param->dual_filter = false;       /* 是否开启滤波器双径滤波，开启后增加滤波器鲁棒性 */
```

PF子模块参数设置

PF模块通过rkaudio_pf_param_init()函数配置，主要参数如下：

```
param->fast_filter_len = 4;      /* PF滤波器长度，实际可覆盖回声长度为fast_filter_len *
帧长 */
param->delay_len = 1;            /* 回采延迟长度，即对回采进行延迟，实际延迟长度为delay_len
* 帧长 */
param->start_freq = 0;           /* PF处理起始频率，该值必须大于0 */
param->end_freq = 300;           /* PF处理结束频率，该值必须小于采样率 */
param->alpha = 0.995f;           /* PF平滑系数，该值越小滤波器更新速度越快 */
```

AES模块参数设置

AES模块用于非线性处理过程，该模块通过rkaudio_aes_param_init()配置。

```
param->Beta_Up           /* 上升速度，该值越大则非线性抑制越强，建议0.001-0.01之间  */
param->Beta_Down         /* 下降速度，该值越大则非线性抑制越弱，建议0.001-0.01之间  */
param->Beta_Up_Low       /* 低频段上升速度，特性以及取值同Beta_Up */
param->Beta_Down_Low     /* 低频段下降速度，特性以及取值同Beta_Down */
param->low_freq          /* 低频分段 */
param->high_freq         /* 高频分段 */
param->start_freq = 0;   /* AES起始作用频率，该值必须大于0 */
param->end_freq = 400;   /* AES结束作用频率，该值必须小于采样率*/
param->THD_Flag          /* 置1开启谐波失真抑制，置0则关闭 */
param->HARD_Flag         /* 置1开启Hard模式，对于回声消除抑制更强，但也有可能造成语音过消 */
int i, j;

/**
 * 3列分别对应低频分段、中频分段、高频分段（由上面的low_freq以及high freq设置
 * 第一行表示各频段残留回声的比例最小值，一般建议取值范围[1.0f, 5.0f]，设置越大，回声消除效果越强
 * 第二行表示各频段消除残留回声的比例，一般建议取值范围[1.0f, 3.0f]，设置越大，回声消除效果越强
 */
param->LimitRatio[2][3]

/**
 * 谐波抑制频段，最多设置4个频段，不需要时该行可以全置0
 * 每一行表示一个频段，第一列表示频段起始点，第二列表示频段终止点。取值范围均为[0,1/2*sample rate]
 */
param->ThdSplitFreq[4][2]

/**
 * 谐波抑制程度，对应上表的四个频段，配置的对应消除比例，当上面有全0行时，该表对应行的设置也将无效
 * 从第一列到最后一列分别表示频段内有可能成为二次到十一次谐波时消除相应基频的比例，一般建议取值范围
 * [0.001f,1.0f]，设置越大，消除效果越强
 */
param->ThdSupDegree[4][10]

/**
 * 配置为Hard模式的频段以及计算平均增益的频段，第一列表示起始，第二列表示终止，取值范围均为
 * [0,1/2*sample rate]，配置为Hard模式的频段最多设置4个，最后一行为计算平均增益的频段，
 * 该频段将在下一个参数HardThreshold中进行使用
 */
param->HardSplitFreq[5][2]

/**
 * Hard模式开启阈值，4个阈值分别对应上个参数配置的4个频段
 * 使用平均增益频段和该参数配置的阈值进行对比，当平均增益低于阈值时，对应的频段会开启Hard模式
 */
param->HardThreshold[4]
```

被动抑制模块参数设置

其中各子模块开关主要通过**model_en**控制，主要包括：*RKAUDIO_EN_NOTCH*、*RKAUDIO_EN_SHIFT*、*RKAUDIO_EN_VC*。具体可看下面代码注释。

NOTCH子模块参数设置

NOTCH模块通过rkaudio_notch_param_init()函数配置，主要参数如下：

```
param->Div_Num = 3;                                     /* 划分频带数，如3，将总体需要检测和抑制频率划分为低、中、高三个频带  */
param->Freq_Div = Freq_Div;                             /* 划分频点，长度为Div_Num + 1,如示例，低频100-1400，中频1400-4900，高频4900-14000，单位Hz  */
param->Power_Threshold = Power_Threshold;               /* 啸叫检测绝对阈值，频点能量超过该值则判断为啸叫，默认-1不使能（非专业人士不推荐开启）  */
param->PHarmonic_Threshold = PHarmonic_Threshold;       /* 啸叫检测谐波比率=基频/谐波，当频点能量除以谐波能量超过该值判断为啸叫，默认-1不使能，单位dB*/
param->PAverage_Threshold = PAverage_Threshold;         /* 啸叫检测平均比率=基频/平均能量，当频点能量除以平均能量超过该值判断为啸叫，默认-1不使能，单位dB  */
param->PNeibor_Threshold = PNeibor_Threshold;           /* 啸叫检测区域比率=基频/区域频点能量，当频点能量除以剔除该频点的周边能量超过该值判断为啸叫，默认-1不使能，单位dB  */
param->SFM_Threshold = SFM_Threshold;                   /* 语音平坦度检测，平坦度由SFM_LEN帧的能量梯度计算，当平坦度大于该值则判断为啸叫，默认-1不使能 */
param->Fund_Correct = Fund_Correct;                      /* 基频能量检测， 当频点能量/基频能量大于该值threshold时判断为啸叫，默认-1不使能，具体参数详情见示例 */
param->SFM_LEN = 5;                                     /* SFM计算帧长  */
param->IMSD_LEN = 6;                                    /* 预留参数  */
param->Persist_Frame = 5;                               /* 持续检测帧长，单一频点持续检测啸叫状态大于Persist_Frame时，施加总体gain  */
param->Detect_Frame = 3;                                /* 持续检测帧长，单一频点持续检测啸叫状态大于Detect_Frame时，施加陷波  */
param->SFreq_Num = 20;                                  /* 最大同时施加的陷波数量  */
param->gain = -10;                                      /* 持续啸叫总体gain，当单一频点持续检测啸叫状态帧长大于Persist_Frame时，施加总体gain  */
```

NOTCH模块参数示例如下：

```
static int Freq_Div[Div_Num + 1]             = { 100, 1400, 4900, 14000 };
static int Power_Threshold[Div_Num]         = { -1, -1, -1 };
static int PHarmonic_Threshold[Div_Num]      = { -1, -1, -1};
static int PAverage_Threshold[Div_Num]       = { -1, -1, -1};
static int PNeibor_Threshold[Div_Num]        = { 22, 15, 10 };
static int SFM_Threshold[Div_Num]            = { 390, 100, 30 };
static int Fund_Correct[5]                   = {1, 6, 10, 100, 10};
Fund_Correct固定长度5，其中fund_start和fund_end为基频检测起始频点，fund_correct_start和
fund_correct_end为基频检测施加频点，fund_correct_threshold为能量/基频能量阈值，大于该值为
啸叫频点。
fund_start = Fund_Correct[0];
fund_end = Fund_Correct[1];
fund_correct_start = Fund_Correct[2];
fund_correct_end = Fund_Correct[3];
fund_correct_threshold = Fund_Correct[4];
```

SHIFT子模块参数设置

SHIFT模块通过rkaudio_shift_param_init()函数配置，主要参数如下：

```
param->Filter_Tap = 151;         /* must be odd, such 59,91,;and response delay is
30,46;tipically 151(which is pre_store using LS method) */
param->Shift_Freq = -2.5;        /* 频移Hz，正数频率加，负数频率减; */
```

VC子模块参数设置

VC模块通过rkaudio_vc_param_init()函数配置，主要参数如下：

```
param->FsRatio = 0.75;           /* freq shift ratio */
param->FreqApply = 1;            /* 0:time domain and may cause vibrate; 1:
frequency domain,can avoid vibrate, but will cause one Frame delay */
```

注意：该模块的频域处理会引入一帧延迟，所有频域处理共享一帧延迟。

辅助模块参数设置

其中各子模块开关主要通过**model_en**控制，主要包括：*RKAUDIO_EN_HPF*、*RKAUDIO_EN_Limiter*、*RKAUDIO_EN_NLP*。具体可看下面代码注释。

HPF子模块参数设置

HPF模块通过rkaudio_hpf_param_init()函数配置，主要参数如下：

```
param->fc = 60;                 /* 高通滤波器截至频率 */
param->bw = 1;
```

LIMIT子模块参数设置

LIMIT模块通过rkaudio_limit_param_init()函数配置，主要参数如下：

```
param->PLimit = PLimit;         /* Limit参数，具体见示例 */
param->graph_x = GraphX;        /* Limit输入dB */
param->graph_y = GraphY;        /* Limit输出dB */
```

```
static float PLimit[6] = { 200, 0, 200, 0, 0, 5 };
static float GraphX[5] = { -35, -25, -10, -2, 6 };
static float GraphY[5] = { -40, -25, -10, -3, 0 };
pParameter[kCompressorRmstc] = 200;          /* Limit参数，Rmstc能量追踪速率，单位ms */
pParameter[kCompressorHoldTime] = 0;         /* Limit参数，HoldTime当增益下降时，增益保持不变时间，单位ms */
pParameter[kCompressorDecayTime] = 200;      /* Limit参数，DecayTime增益下降速度，单位ms */
pParameter[kCompressorSoftKnee] = 0;         /* Limit参数，软拐点使能判断 */
pParameter[kCompressorPostGain] = 0;         /* Limit参数，线性增益 */
pParameter[kCompressorTableLen] = 5;         /* Limit参数，table长度，与GraphX和GraphY对齐 */
```

NLP子模块参数设置

NLP模块通过rkaudio_nlp_param_init()函数配置，主要参数如下：

```
param->alpha = 0.3f;                         /* 解相关系数，该值越大去相关越好，推荐0.1-0.3之间 */
```

简要调试说明

客户可根据需求配置主动和被动啸叫抑制，两者可以结合使用也可以单独使用。

主被动啸叫抑制推荐配置

主动啸叫抑制推荐开启：RKAUDIO_EN_AF | RKAUDIO_EN_NLP | RKAUDIO_EN_Limiter | RKAUDIO_EN_HPF | RKAUDIO_EN_SHIFT 该配置总体语音保真度高，但是AF模块需要仔细处理延迟和滤波器长度，并且如果PA线性度太差，该模块可能会失效。

纯被动啸叫抑制推荐配置

被动啸叫抑制推荐开启：RKAUDIO_EN_NOTCH | RKAUDIO_EN_Limiter | RKAUDIO_EN_HPF | RKAUDIO_EN_SHIFT

该配置适应性较高，不需要强调PA线性度。

NOTCH调试过程中，推荐使用PHarmonic、PAverage、PNeibor_Threshold、SFM_Threshold、Fund_Correct中的两个进行有机组合。

需要主要，调试过程中需要精细调整每个参数，建议先将所有模块置-1，单一模块开启后，需要调试参数测不同工况的抑制和过消等。