H2 API
编程指南

V2.0.18
2026/05/22

HAROGIC 海得逻捷

目录

1. 版本信息 6

2. 概述 7

3. 设备及 API 版本 8

4. 函数类别简介 9

5. API 调用流程 10

5.1 CW 模式 10

5.2 实时流模式 10

5.3 回放模式 10

6. 设备与系统 device 函数 11

6.1 device_list_usb 11

6.2 device_open_usb 11

6.3 device_open_eth 12

6.4 device_close 13

6.5 device_preset 14

6.6 device_query_temperature 14

6.7 device_query_state 15

6.8 device_config_fan 15

6.9 device_query_supply 16

6.10 device_config_gnss 17

6.11 device_query_gnss 17

6.12 device_query_gnss_info 17

6.13 device_query_apiverion 18

6.14 device_config_clock 18

6.15 device_query_clock 19

6.16 device_reset_clock_monitor 20

6.17 device_epoch_to_utc 21

7. 通道设置函数 23

7.1 channel_preset 23

7.2 channel_start 23
7.3 channel_stop 24
7.4 channel_trigger_bus 24
7.5 channel_config_trigger 24
7.6 channel_query_trigger 25
7.7 channel_config_trigger_out 25
7.8 channel_query_trigger_out 26
7.9 channel_config_lo_mode 27
7.10 channel_query_lo_mode 28
8. 发射设置函数 29
8.1 tx_clear_waveform 29
8.2 tx_download_waveform 29
8.3 tx_config_ffm 30
8.4 tx_query_ffm 30
8.5 tx_config_fscan 31
8.6 tx_query_fscan 32
8.7 tx_config_lscan 33
8.8 tx_query_lscan 34
8.9 tx_config_output 35
8.10 tx_query_output 35
8.11 tx_config_cw 36
8.12 tx_config_stream 37
8.13 tx_send_stream 37
8.14 tx_config_playback 39
9. 调制设置函数（选件） 42
9.1 mod_open 42
9.2 mod_close 42
9.3 mod_init_digitalmod 43
9.4 mod_init_am 43

9.5 mod_init_fm 43
9.6 mod_init_dsss 44
9.7 mod_init_pulse 44
9.8 mod_init_multitone 45
9.9 mod_init_stepsweep 45
9.10 mod_init_rampsweep 45
9.11 mod_init_ofdm 46
9.12 mod_init_awgn 46
9.13 mod_config_digitalmod 47
9.14 mod_config_am 47
9.15 mod_config_fm 48
9.16 mod_config_dsss 48
9.17 mod_config_pulse 49
9.18 mod_config_multitone 49
9.19 mod_config_stepsweep 50
9.20 mod_config_rampsweep 50
9.21 mod_config_ofdm 51
9.22 mod_config_awgn 51
9.23 mod_generate_stream 52
10. 结构体变量 55
10.1 device_info 55
10.2 supply_info 55
10.3 eth_setting 55
10.4 gnss_setting 56
10.5 gnss_info 56
10.6 channel 56
10.7 tx_trigger 56
10.8 trigger_out 57
10.9 digitalmod_profile 57

10.10 am_profile 58
10.11 fm_profile 58
10.12 dsss_profile 58
10.13 pulse_profile 58
10.14 multitone_profile 59
10.15 stepsweep_profile 59
10.16 rampsweep_profile 59
10.17 ofdm_profile 59
10.18 awgn_profile 60
11. 枚举变量 61
11.1 state 61
11.2 fan_mode 61
11.3 eth_interface_type 61
11.4 referenceclock_source 61
11.5 lomode 61
11.6 channel_type 61
11.7 trigger_action 61
11.8 trigger_edge 62
11.9 trigger_source 62
11.10 mod_digitalmod_type 62
11.11 mod_filter_type 63
11.12 mod_shape_type 63
附录 Appendix 1：API 返回值索引 64

1. 版本信息

版本更新说明表

版本号	内容	时间
V2.0.18	增加：调制设置函数（选件）章节	2026-5-22
V2.0.17	增加：device_query_state 函数	2026-4-20
V2.0.12	增加： device_open_eth、 device_config_fan、 device_config_clock、 device_query_clock、 device_reset_clock_monitor、 channel_config_trigger_out、 channel_query_trigger_out、 channel_config_lo_mode、 channel_query_lo_mode、 tx_config_lscan 和 tx_query_lscan 函数	2026-4-17
V2.0.10	增加： device_list_usb、 device_query_supply device_config_gnss、 device_query_gnss、 device_query_gnss_info、 device_epoch_to_utc、 tx_config_fscan、 tx_query_fscan 和 tx_config_stream 函数	2026-3-20
V1.0	初始版本	2026-1-10

2. 概述

此API系统是基于C编写的动态链接库，用于对设备进行编程开发。信号源主要有以下三种工作模式：

连续波模式：	该模式下，设备发送连续波，用户无需向设备下发数据。
实时流模式：	该模式下，用户可通过USB总线向设备发送IQ波形数据，设备接收数据并实时播放。当用户停止下发后，设备将下发的所有数据播放完成后停止播放。实时数据的采样率（上限62.5MHz）和传输速率受限于数据接口的物理带宽，数据大小不受限。
回放模式：	该模式采用预存储波形播放机制，用户可先将IQ波形数据下载至设备存储器中，后续通过指定波形的索引、循环播放次数、采样率和播放的波形数量实现多波形播放。由于所有波形数据已预先存储，无需依赖高速实时数据传输，即可实现最大模拟带宽的波形输出。需注意的是，可存储的波形数据总量受限于设备的存储器容量（上限125 MB）。

3. 设备及 API 版本

系统是多个软固件的联合运行，在本系统中，涉及的软固件包括：设备主控固件（MFW/MCU）、FPGA固件（FFW）、总线固件（Bus）、AGU固件版本（AGU）及应用程序接口（API）。

版本匹配原则

为确保系统稳定运行，上述各组件版本必须遵循严格的**基线版本对齐**原则。可参照下表进行版本的核对与部署。

表格 1 软固件基线版本对应表

API	FPGA	MCU	Bus
2.0.18	2.0.12	2.0.19	2.0.1
2.0.17	2.0.11	2.0.13	2.0.1

注意：1）为避免系统异常，切勿将不同基线版本中的软固件交叉使用，请务必严格按照上表所列版本进行统一部署或升级。

2）确保实际使用的API版本与手册封面所标识的对应，以免出现手册接口说明与实际功能不符的情况。

4. 函数类别简介

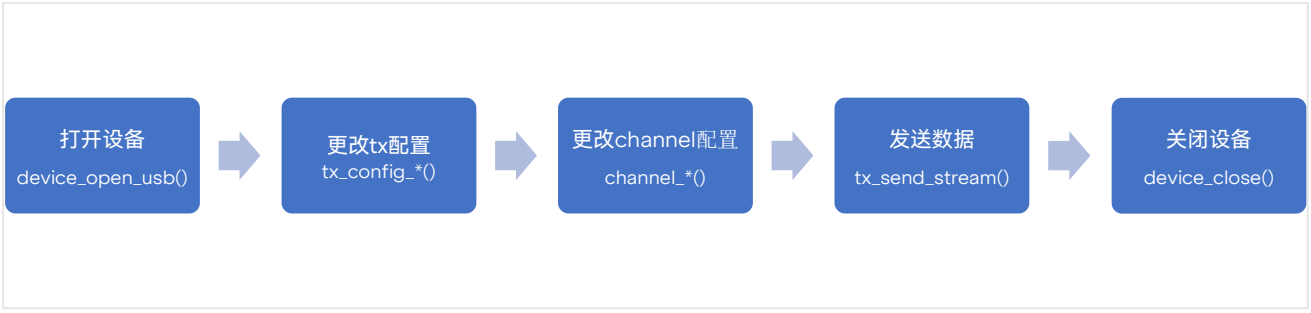
表格 2 API 函数类别

函数类别	说明
设备与系统 device	全局功能，可在任意工作模式下调用此类别下的函数。该类别包括对设备打开、关闭、全局性设置、获取设备信息和获取设备状态等功能。
通道设置 channel	通道级控制，用于管理设备内部独立的信号通路。可实现对指定通道的配置、使能、参数设置与状态查询，是进行多通道同步或独立控制的基础。
发射设置 tx	发射链路控制。此类别函数用于配置发射参数、控制信号发射流程、管理发射数据及查询发射状态，是信号源功能的核心。

5. API 调用流程

5.1 CW 模式

5.2 实时流模式



5.3 回放模式

6. 设备与系统 device 函数

在调用任何与硬件关联的 API 之前，必须首先调用 device_open_usb 初始化设备。完成业务操作后，应调用 device_close 以关闭设备并释放分配的内存资源。

6.1 device_list_usb

int device_list_usb(device_info* **devInfo,** uint32_t* **count** **)**	
功能描述	
枚举当前连接至主机的 USB 设备列表。	
兼容性	2.0.2及之后版本支持
参数说明	
[out] **devInfo**	返回当前所有已连接设备的基本信息。有效信息包含 model 和 uid，详见device_info结构体定义。
[out] **count**	当前连接至主机的 USB 设备总数。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	无。
示例	

device_info devInfo;
uint32_t count = 0;
int status = device_list_usb(&devInfo, &count);	

6.2 device_open_usb

int device_open_usb(void** **device,** uint8_t **dnum,** channel **ch_o[],** uint16_t* **chn,** device_info* **dinfo** **)**	
功能描述	
初始化设备连接。本函数根据启动配置建立底层通信并枚举可用通道。在执行任何后续 API 操作前，必须先调用此函数。成功时返回设备句柄、通道列表、通道数量及设备基本信息	
兼容性	2.0.1及之后版本支持
参数说明	

[out] **device**	返回设备句柄。在调用后续 API 时，使用该句柄引用目标设备。
[in] **dnum**	指定要打开的设备索引。当连接多台设备时，使用该索引选择目标设备。索引从 **0** 开始递增。
[out] **ch_o**	返回设备支持的通道数组，详见channel结构体定义。
[out] **chn**	返回通道总数。
[out] **dinfo**	返回设备基本信息，详见device_info结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在调用任何其他函数前执行。此函数仅需在初始化阶段调用一次。后续 API 操作将基于此函数返回的设备句柄执行。对于所有成功的 **device_open_usb** 调用，必须在模块使用结束后调用 **device_close** 以释放资源。
示例	请参考tx_config_cw()函数相关示例。

6.3 device_open_eth

```
int device_open_eth(
    void** device,
    eth_setting settings,
    channel ch_o[],
    uint16_t* chn,
    device_info* dinfo
)
```

功能描述	
初始化以太网设备连接。本函数根据启动配置建立底层通信并枚举可用通道。在执行任何后续 API 操作前，必须先调用此函数。成功后返回设备句柄、通道列表、通道数量及设备基本信息。	
兼容性	2.0.11及之后版本支持
参数说明	
[out] **device**	返回设备句柄。在调用后续 API 时，使用该句柄引用目标设备。
[in] **settings**	以太网设备启动配置，详见eth_setting结构体定义。
[out] **ch_o**	返回设备支持的通道数组，详见channel结构体定义。
[out] **chn**	返回通道总数。
[out] **dinfo**	返回设备基本信息，详见device_info结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在调用任何其他函数前执行。此函数仅需在初始化阶段调用一次。后续 API 操作将基于此函数返回的设备句柄

	（Handle）执行。对于所有成功的 device_open_eth 调用，必须在模块使用结束后调用 device_close 以释放资源。
示例	

int status = 0;
void* device = nullptr;
device_info dinfo;
channel ch[8]; uint16_t chn = 0;
eth_setting settings;
settings.eth_interface = ETH_STANDARD;
settings.eth_is_ipv6 = 0;
settings.eth_ip_address[0] = 192;
settings.eth_ip_address[1] = 168;
settings.eth_ip_address[2] = 1;
settings.eth_ip_address[3] = 100;
settings.eth_remote_port = 5000;
settings.eth_read_timeout = 5000;
status = device_open_eth(&device, settings, ch, &chn, &dinfo);
status = device_close(&device);	

6.4 device_close

int device_close(void device)**	
功能描述	
关闭设备连接并释放相关资源。	
兼容性	1.0.1及之后版本支持
参数说明	
[in] **device**	设备句柄。关闭后该句柄将失效。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	仅在程序执行结束时调用此函数。执行后，设备连接将关闭且相关资源被释放。如需再次操作设备，必须重新调用 device_open_* 以重新建立连接并初始化设备。
示例	请参考tx_config_cw()函数相关示例。

6.5 device_preset

int device_preset(void device)**	
功能描述	
执行设备复位（包括所有通道的全局复位）。	
兼容性	1.0.1及之后版本支持
参数说明	
[in] **device**	设备句柄。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
status = device_preset(&device);
status = device_close(&device);
```

6.6 device_query_temperature

int device_query_temperature(**void** device,** **float* temp_o** **)**	
功能描述	
查询设备当前的温度。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **temp_o**	返回设备温度，单位：摄氏度。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
int status = -1;
```

```
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
float temp_o = 0;
status = device_query_temperature(&device, &temp_o);
status = device_close(&device);
```

6.7 device_query_state

int device_query_state(void device)**	
功能描述	
查询设备当前状态	
兼容性	1.0.1及之后版本支持
参数说明	
[in] **device**	设备句柄。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。

6.8 device_config_fan

int device_config_fan(**void** device,** **fan_mode mode,** **float auto_threshold** **)**	
功能描述	
配置风扇运行参数。	
兼容性	2.0.12及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **mode**	设置风扇工作模式。详见fan_mode枚举定义。
[in] **auto_threshold**	设置自动模式下的温度阈值。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数
示例	

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
fan_mode mode = FAN_AUTO;
float auto_threshold = 40.0;
status = device_config_fan(&device, mode, auto_threshold);
status = device_close(&device);
```

6.9 device_query_supply

```
int device_query_supply(
    void** device,
    supply_info* supplyInfo_o
)
```

功能描述	
查询设备当前的供电状态及电源参数。	
兼容性	2.0.10及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **supplyInfo_o**	返回设备的电压与电流参数。详见supply_info结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。

示例

```
int status = -1; void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0; uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
supply_info supplyInfo_o;
status = device_query_supply(&device, &supplyInfo_o);
status = device_close(&device);
```

6.10 device_config_gnss

int device_config_gnss(void** device, gnss_setting gnss)	
功能描述	
配置设备的GNSS运行参数。	
兼容性	2.0.7及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **gnss**	GNSS配置参数。详见gnss_setting结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考device_epoch_to_utc()函数相关示例。

6.11 device_query_gnss

int device_query_gnss(void** device, gnss_setting* gnss_o)	
功能描述	
查询设备当前的GNSS配置。	
兼容性	2.0.7及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **gnss**	返回当前的GNSS配置参数。详见gnss_setting结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考device_epoch_to_utc()函数相关示例。

6.12 device_query_gnss_info

int device_query_gnss_info(void** device, gnss_info* gnss_o)
功能描述
查询设备当前的GNSS信息。

兼容性	2.0.1及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **gnss_o**	返回设备的GNSS信息。详见gnss_info结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考device_epoch_to_utc()函数相关示例。

6.13 device_query_apiverion

uint32_t device_query_apiverion()	
功能描述	
返回当前库对应的 API 版本信息。	
兼容性	1.0.1及之后版本支持
返回值	API 版本号。采用主版本，子版本，修订版本表示。 bit[31..16]表示主版本，bit[15..8]表示子版本，bit[7..0]表示修订。
调用约束	无。
示例	

```
uint32_t verion = device_query_apiverion();
int major = 0, minor = 0, rev = 0;
major = (verion >> 16) & 0xffff;
minor = (verion >> 8) & 0xff;
rev = verion & 0xff;
```

6.14 device_config_clock

```
int device_config_clock(
    void** device,
    referenceclock_source source,
    double fref,
    state out
)
```

功能描述	
配置设备的参考时钟源。	
兼容性	2.0.11及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **source**	设置参考时钟源。详见referenceclock_source结构体定义。

[in] **fref**	设置参考时钟频率，单位 Hz。
[in] **out**	设置参考时钟输出的启用状态。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
referenceclock_source source = REFCLKSOURCE_EXTERNAL;
double fref = 10e6;
state out = STATE_OFF;
status = device_config_clock(&device, source, fref, out);
state locked_o;
status = device_query_clock(&device, &source, &fref, &out, &locked_o);
status = device_close(&device);
```

6.15 device_query_clock

```
int device_query_clock(
    void** device,
    referenceclock_source* source_o,
    double* fref_o,
    state* out_o,
    state* locked_o
)
```

功能描述	
查询设备当前的参考时钟配置状态。	
兼容性	2.0.11及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **source_o**	返回的参考时钟源。详见referenceclock_source结构体定义。
[out] **fref_o**	返回参考时钟频率，单位 Hz。
[out] **out_o**	返回参考时钟输出的启用状态。

[out] **locked_o**	返回参考时钟监控的锁定状态（仅在状态为“锁定”时，获取的当前实际参考时钟频率才有效）
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考device_config_clock()函数相关示例。

6.16 device_reset_clock_monitor

int device_reset_clock_monitor(void device)**	
功能描述	
复位参考时钟监控。	
兼容性	2.0.11及之后版本支持
参数说明	
[in] **device**	设备句柄。关闭后该句柄将失效。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
status = device_reset_clock_monitor(&device);
status = device_close(&device);
```

6.17 device_epoch_to_utc

void device_epoch_to_utc(**uint64_t ns_sinceepoch,** **int16_t* year,** **int16_t* mon,** **int16_t* day,** **int16_t* hour,** **int16_t* min,** **int16_t* sec,** **int16_t* ms,** **int16_t* us,** **int16_t* ns** **)**	
功能描述	
将Unix Epoch纳秒时间戳 转换为可读的UTC 时间格式。	
兼容性	2.0.7及之后版本支持
参数说明	
[in] **ns_sinceepoch**	自Unix Epoch以来的累计时间。单位：ns。
[out] **year**	返回年份（如 2026）。
[out] **mon**	返回月份（1 ~ 12）。
[out] **day**	返回日期（1 ~ 31）。
[out] **hour**	返回小时（0 ~ 23）。
[out] **min**	返回分钟（0 ~ 59）。
[out] **sec**	返回秒数（0 ~ 59）。
[out] **ms**	返回毫秒数（0 ~ 999）。
[out] **us**	返回微秒数（0 ~ 999）。
[out] **ns**	返回纳秒数（0 ~ 999）。
返回值	无。
调用约束	必须在GNSS 状态为“锁定”且执行 device_query_gnss_info 获取最新授时数据后调用此函数。
示例	

int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0; uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
gnss_setting gnss;	

```
gnss.antenna = 1;
status = device_config_gnss(&device, gnss);
status = device_query_gnss(&device, &gnss);
gnss_info gnss_info_o;
status = device_query_gnss_info(&device, &gnss_info_o);
int16_t year = 0, mon = 0, day = 0, hour = 0, min = 0, sec = 0, ms = 0, us = 0, ns = 0;
device_epoch_to_utc(gnss_info_o.ns_sinceepoch, &year, &mon, &day, &hour, &min, &sec,
&ms, &us, &ns);
status = device_close(&device);
```

7. 通道设置函数

7.1 channel_preset

int channel_preset(channel* ch)	
功能描述	
对多个指定通道执行复位操作。此操作仅完成状态重置，不会触发信号输出。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道数组。详见channel结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行 device_open_* 初始化之后调用此函数。
示例	

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
status = channel_preset(ch);
status = device_close(&device);
```

7.2 channel_start

int channel_start(channel* ch)	
功能描述	
启动多通道运行；在修改射频（RF）或基带（Baseband）相关参数后，必须调用此函数以使配置生效并启动通道。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道数组，详见channel结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

7.3 channel_stop

int channel_stop(channel* ch)	
功能描述	
停止多个指定通道的运行。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道数组，详见channel结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

7.4 channel_trigger_bus

int channel_trigger_bus(channel* ch)	
功能描述	
向多个指定通道发送一次 总线触发指令。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道数组，详见channel结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

7.5 channel_config_trigger

int channel_config_trigger(channel* ch, const tx_trigger* trg)	
功能描述	
配置指定通道的触发参数。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体定义。
[in] **trg**	触发配置参数，详见tx_trigger结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。

调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

7.6 channel_query_trigger

int channel_query_trigger(**channel* ch,** **tx_trigger* trg_o** **)**
功能描述
查询指定通道的当前触发配置。

兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体的定义。
[out] **trg_o**	触发配置返回信息，详见tx_trigger结构体的定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

7.7 channel_config_trigger_out

int channel_config_trigger_out(**channel* ch,** **const trigger_out* trg** **)**
功能描述
配置指定通道的触发输出参数。

兼容性	2.0.11及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体的定义。
[in] **trg**	触发输出配置参数，详见trigger_out结构体的定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
```

```
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
double start = 1.0e9;
double stop = 2.0e9;
double step = 100e6;
float level = 0.0f;
float dwell = 1e-4;
status = tx_config_fscan(ch, start, stop, step, level, dwell);
tx_trigger trg;
trg.source = TRIGGER_SOURCE_BUS;
trg.action = TRIGGER_ACTION_SWEEP;
trg.count = -1;
status = channel_config_trigger(ch, &trg);
trigger_out trgout;
trgout.enable = STATE_ON;
trgout.action = TRIGGER_ACTION_HOP;
trgout.edge = TRIGGER_EDGE_RISING;
status = channel_config_trigger_out(ch, &trgout);
status = channel_query_trigger_out(ch, &trgout);
status = tx_config_cw(ch);
status = tx_config_output(ch, STATE_ON, STATE_OFF);
status = channel_start(ch);
status = channel_trigger_bus(ch);
status = device_close(&device);
```

7.8 channel_query_trigger_out

```
int channel_query_trigger_out(
    channel* ch,
    trigger_out* trg_o
)
```

功能描述	
查询指定通道当前的触发输出配置。	
兼容性	2.0.11及之后版本支持
参数说明	

[in] **ch**	目标通道，详见channel结构体的定义。
[out] **trg_o**	返回触发输出配置参数，详见trigger_out结构体的定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考channel_config_trigger_out()函数相关示例。

7.9 channel_config_lo_mode

int channel_config_lo_mode(

channel* ch,
lomode mode
)

功能描述

配置指定通道的本振模式。

兼容性	2.0.11及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体的定义。
[in] **mode**	本振模式配置，详见lomode结构体的定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。

示例

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
lomode mode = LOMODE_LOWPHASENOISE;
status = channel_config_lo_mode(ch, mode);
status = channel_query_lo_mode(ch, &mode);
status = device_close(&device);
```

7.10 channel_query_lo_mode

int channel_query_lo_mode(channel* ch, lomode* mode_o)	
功能描述	
查询指定通道当前的本振 模式配置。	
兼容性	2.0.11及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体的定义。
[out] **mode_o**	返回本振模式配置，详见lomode结构体的定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考channel_config_lo_mode()函数相关示例。

8. 发射设置函数

8.1 tx_clear_waveform

int tx_clear_waveform(void** device)	
功能描述	
擦除设备内部存储器中的所有波形数据。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **device**	设备句柄。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_playback()函数相关示例。

8.2 tx_download_waveform

int tx_download_waveform(void** device, int16_t iq[], uint32_t points, uint16_t* waveform_o)	
功能描述	
将指定的波形样本数据下传至设备内部存储器。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **iq**	IQ 数据数组。
[in] **points**	IQ 点数对。
[out] **waveform_o**	波形序号。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_playback()函数相关示例。

8.3 tx_config_ffm

```
int tx_config_ffm(
    channel* ch,
    double fc,
    float level
)
```

功能描述	
配置指定通道的频率与幅度参数（FFM模式）。	
兼容性	1.0.1及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体定义。
[in] **fc**	中心频率。
[in] **level**	输出幅度。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

8.4 tx_query_ffm

```
int tx_query_ffm(
    channel* ch,
    double* fc_o,
    float* level_o
)
```

功能描述	
查询指定通道当前的频率与幅度参数 (FFM 模式)。	
兼容性	1.0.1及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体的定义。
[out] **fc_o**	返回中心频率。
[out] **level_o**	返回输出幅度。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

8.5 tx_config_fscan

```
int tx_config_fscan(
    channel* ch,
    double start,
    double stop,
    double step,
    float level,
    float dwell
)
```

功能描述	
配置指定通道的频率扫描参数。在固定电平下，以 Step 为步进，执行从 Start 频率到 Stop 频率的递增扫描。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体定义。
[in] **start**	扫描起始频率。
[in] **stop**	扫描终止频率。
[in] **step**	扫描步进频率。
[in] **level**	输出幅度。
[in] **dwell**	频点驻留时间 (Dwell Time)。单位：秒。 在TRIGGER_ACTION_HOP模式下，驻留时间设置不生效；频率切换仅由外部触发信号驱动。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行device_open_*初始化之后调用此函数。

示例

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
double start = 1.0e9;
double stop = 2.0e9;
double step = 100e6;
float level = 0.0f;
```

```
float dwell = 1e-4;
status = tx_config_fscan(ch, start, stop, step, level, dwell);
status = tx_query_fscan(ch, &start, &stop, &step, &level, &dwell);
tx_trigger trg;
trg.source = TRIGGER_SOURCE_BUS;
trg.action = TRIGGER_ACTION_SWEEP;
trg.count = -1;
status = channel_config_trigger(ch, &trg);
status = tx_config_cw(ch);
status = tx_config_output(ch, STATE_ON, STATE_OFF);
status = channel_start(ch);
status = channel_trigger_bus(ch);
status = device_close(&device);
```

8.6 tx_query_fscan

```
int tx_query_fscan(
    channel* ch,
    double* start_o,
    double* stop_o,
    double* step_o,
    float* level_o,
    float* dwell_o
)
```

功能描述

查询指定通道当前的频率扫描参数。

兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体定义。
[out] **start_o**	返回扫描起始频率。
[out] **stop_o**	返回扫描终止频率。
[out] **step_o**	返回扫描步进频率。
[out] **level_o**	返回输出幅度。
[out] **dwell_o**	返回单个频点驻留时间（秒）。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_fscan()函数相关示例。

8.7 tx_config_lscan

```
int tx_config_lscan(
    channel* ch,
    double fc,
    float level_start,
    float level_stop,
    float level_step,
    float dwell
)
```

功能描述

配置指定通道的幅度扫描参数。在固定中心频率 fc 下，以 Level Step 为步进，执行从 Level Start 到 Level Stop 的递增扫描。

兼容性	2.0.11及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体定义。
[in] **fc**	中心频率。
[in] **level_start**	扫描起始幅度。
[in] **level_stop**	扫描终止幅度。
[in] **level_step**	扫描步进幅度。
[in] **dwell**	幅度点驻留时间。单位：秒。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。

示例

```
int status = -1;
void *device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
double fc = 1.0e9;
float level_start = -20;
float level_stop = -10;
float level_step = 1;
float dwell = 1e-4;
status = tx_config_lscan(ch, fc, level_start, level_stop, level_step, dwell);
status = tx_query_lscan(ch, &fc, &level_start, &level_stop, &level_step, &dwell);
```

```
tx_trigger trg;
trg.source = TRIGGER_SOURCE_BUS;
trg.action = TRIGGER_ACTION_SWEEP;
trg.count = -1;
status = channel_config_trigger(ch, &trg);
status = tx_config_cw(ch);
status = tx_config_output(ch, STATE_ON, STATE_OFF);
status = channel_start(ch);
status = channel_trigger_bus(ch);
status = device_close(&device);
```

8.8 tx_query_lscan

```
int tx_query_lscan(
    channel* ch,
    double* fc_o,
    float* level_start_o,
    float* level_stop_o,
    float* level_step_o,
    float* dwell_o
)
```

功能描述

查询指定通道当前的幅度扫描参数。

兼容性	2.0.11及之后版本支持
参数说明	
[in] **ch**	目标通道，详见channel结构体的定义。
[out] **fc_o**	返回中心频率。
[out] **level_start_o**	返回扫描起始幅度。
[out] **level_stop_o**	返回扫描终止幅度。
[out] **level_step_o**	返回扫描步进幅度。
[out] **dwell_o**	幅度点驻留时间返回信息。单位：秒。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_lscan()函数相关示例。

8.9 tx_config_output

int tx_config_output(channel* ch, state rfout, state mod)	
功能描述	
配置射频与调制输出状态。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道配置数组，详见channel结构体的定义。
[in] **rfout**	射频输出状态配置，详见state枚举的定义。
[in] **mod**	调制输出状态配置，详见state枚举的定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

8.10 tx_query_output

int tx_query_output(channel* ch, state* rfout_o, state* mod_o)	
功能描述	
查询多个通道的射频和调制输出状态。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道配置数组，详见channel结构体定义。
[out] **rfout_o**	返回射频输出状态，详见state枚举定义。
[out] **mod_o**	返回调制输出状态，详见state枚举定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_config_cw()函数相关示例。

8.11 tx_config_cw

int tx_config_cw(channel* ch)	
功能描述	
多通道连续波 (CW) 发射配置。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道配置数组，详见channel结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
double fc = 1.0e9;
float level = 0.0f;
status = tx_config_ffm(ch, fc, level);
status = tx_query_ffm(ch, &fc, &level);
tx_trigger trg;
trg.source = TRIGGER_SOURCE_BUS;
status = channel_config_trigger(ch, &trg);
status = channel_query_trigger(ch, &trg);
status = tx_config_cw(ch);
status = tx_config_output(ch, STATE_ON, STATE_OFF);
state rfout_o, mod_o;
status = tx_query_output(ch, &rfout_o, &mod_o);
status = channel_start(ch);
status = channel_trigger_bus(ch);
std::this_thread::sleep_for(std::chrono::seconds(10));
status = channel_stop(ch);
status = device_close(&device);
```

8.12 tx_config_stream

int tx_config_stream(channel* ch, double sample_rate)	
功能描述	
配置指定通道的流播放参数。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道配置数组，详见channel结构体定义。
[in] **sample_rate**	通道采样率。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	请参考tx_send_stream()函数相关示例。

8.13 tx_send_stream

int tx_send_stream(channel* ch, int16_t iq[], uint32_t points)	
功能描述	
对指定通道发送IQ流实时播放。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道配置数组，详见channel结构体定义。
[in] **iq**	IQ 数据数组。
[in] **points**	IQ 点数对。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
// 仿真波形：生成正弦波
std::vector<int16_t> sim_generatewaveform_sine()
{
    constexpr uint32_t TOTAL_INT16_POINTS = 500000;
    constexpr uint32_t IQ_POINTS = TOTAL_INT16_POINTS / 2;
```

```
    constexpr int16_t AMP = 30000;
    constexpr uint32_t CYCLES = 100;
    const double twoPi = 2.0 * 3.141592654;
    std::vector<int16_t> iq;
    iq.reserve(TOTAL_INT16_POINTS);

    for (uint32_t n = 0; n < IQ_POINTS; ++n) {
        double phase = twoPi * CYCLES * n / IQ_POINTS;
        iq.push_back(static_cast<int16_t>(AMP * std::cos(phase)));
        iq.push_back(static_cast<int16_t>(AMP * std::sin(phase)));
    }
    return iq;
}

int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
double samplerate = 50e6;
std::vector<int16_t> data;
data = sim_generatewaveform_sine();
double fc = 1.0e9;
float level = 0.0f;
status = tx_config_ffm(ch, fc, level);
tx_trigger trg;
trg.source = TRIGGER_SOURCE_BUS;
status = channel_config_trigger(ch, &trg);
status = tx_config_stream(ch, samplerate);
status = tx_config_output(ch, STATE_ON, STATE_ON);
status = channel_start(ch);
status = channel_trigger_bus(ch);
for (int i = 0; i < 100000; ++i) {
```

```
    status = tx_send_stream(ch, data.data(), data.size() / 2);
    if (status != STATUS_NOERROR) {
        break;
    }
}
status = device_close(&device);
```

8.14 tx_config_playback

```
int tx_config_playback(
    channel* ch,
    const int16_t waveform[],
    const int32_t repeat[],
    const double sample_rate[],
    int16_t waveforms
)
```

功能描述	
配置波形回放参数，配置完成后，需通过 channel_trigger_bus 接口触发总线以启动发射。	
兼容性	2.0.1及之后版本支持
参数说明	
[in] **ch**	通道配置数组，详见channel结构体定义。
[in] **waveform**	波形索引数组，指定哪些序号的波形需要被播放。
[in] **repeat**	波形循环计数，指定每个波形段在当前序列中的重复播放次数。
[in] **sample_rate**	每个波形回放的采样率。
[in] **waveforms**	播放的波形数量。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
/* 仿真波形：生成占空比可变的脉冲调制 */
std::vector<int16_t> sim_generatewaveform_pulse(double duty_cycle)
{
  constexpr uint32_t TOTAL_IQ_POINTS = 250000;  // 总 IQ 采样点数
  constexpr uint32_t TOTAL_INT16_POINTS = TOTAL_IQ_POINTS * 2;
  constexpr int16_t I_VALUE = 32766;
  if (duty_cycle < 0.0) duty_cycle = 0.0;
  if (duty_cycle > 1.0) duty_cycle = 1.0;
  uint32_t active_iq = static_cast<uint32_t>(TOTAL_IQ_POINTS * duty_cycle);
```

```
    std::vector<int16_t> iq;
    iq.reserve(TOTAL_INT16_POINTS);
    for (uint32_t i = 0; i < active_iq; ++i) {
        iq.push_back(I_VALUE);  // I
        iq.push_back(0);       // Q
    }
    uint32_t remain_iq = TOTAL_IQ_POINTS - active_iq;
    for (uint32_t i = 0; i < remain_iq; ++i) {
        iq.push_back(0);
        iq.push_back(0);
    }
    return iq;
}
double samplerate = 125e6;
std::vector<int16_t> data;
data = sim_generatewaveform_pulse(0.25)); // 25%占空比脉冲

int status = -1;
void* device = nullptr;
device_info dinfo;
channel ch[8];
uint16_t chn = 0;
uint8_t dnum = 0;
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
double fc = 1.0e9;
float level = 0.0f;
status = tx_config_ffm(ch, fc, level);
tx_trigger trg;
trg.source = TRIGGER_SOURCE_BUS;
status = channel_config_trigger(ch, &trg);
uint16_t waveid = 0;
status = tx_clear_waveform(&device);
status = tx_download_waveform(&device,  data.data(), data.size() / 2, &waveid);
int16_t waveforms = 1;
const int16_t waveform[] = { waveid };
```

```
const int32_t repeat[] = { -1 };
const double srate[] = { samplerate };
status = tx_config_playback(ch, waveform, repeat, srate, waveforms);
status = tx_config_output(ch, STATE_ON, STATE_ON);
status = channel_start(ch);
status = channel_trigger_bus(ch);
std::this_thread::sleep_for(std::chrono::seconds(10));
status = device_close(&device);
```

9. 调制设置函数（选件）

9.1 mod_open

int mod_open(void** device, char version_o[50],)	
功能描述	
开启调制功能，设备打开后，将设备信息结构体传入函数用于校验设备，检查调制库是否存在，注册函数并创建设备对象，打开调制功能。	
兼容性	2.0.18及之后版本支持
参数说明	
[out] **device**	返回设备句柄。在调用后续API时，使用该句柄引用目标设备。
[out] **version_o[50]**	返回调制库版本号
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行device_open_*初始化之后调用此函数。
示例	请参考mod_generate_stream()函数相关示例。

9.2 mod_close

int mod_close(void** device)	
功能描述	
关闭调制功能	
兼容性	2.0.18及之后版本支持
参数说明	
[out] **device**	设备句柄。关闭后该句柄将失效。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	仅在调制程序执行结束前调用此函数。执行后，调制功能将关闭且相关资源被释放。如需再次开启调制功能，必须重新调用mod_open。
示例	请参考mod_generate_stream()函数相关示例。

9.3 mod_init_digitalmod

int mod_init_digitalmod(void** device, digitalmod_profile* digitalmod_profile_o)	
功能描述	
初始化数字调制配置结构体，将结构体中的每个参数都赋初值。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **digitalmod_profile_o**	数字调制配置结构体。详见digitalmod_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。
示例	请参考mod_generate_stream()函数相关示例。

9.4 mod_init_am

int mod_init_am(void** device, am_profile* am_profile_o)	
功能描述	
初始化AM调制配置结构体，将结构体中的每个参数都赋初值。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **am_profile_o**	AM调制配置结构体。详见am_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.5 mod_init_fm

int mod_init_fm(void** device, fm_profile* fm_profile_o)
功能描述
初始化FM调制配置结构体，将结构体中的每个参数都赋初值。

兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **fm_profile_o**	FM调制配置结构体。详见fm_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.6 mod_init_dsss

```
int mod_init_dsss(
    void** device,
    dsss_profile* dsss_profile_o
)
```

功能描述
初始化直接序列扩频配置结构体，将结构体中的每个参数都赋初值。

兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **dsss_profile_o**	直接序列扩频配置结构体。详见dsss_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.7 mod_init_pulse

```
int mod_init_pulse(
    void** device,
    pulse_profile* pulse_profile_o
)
```

功能描述
初始化脉冲调制配置结构体，将结构体中的每个参数都赋初值。

兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **pulse_profile_o**	脉冲调制配置结构体。详见pulse_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.8 mod_init_multitone

int mod_init_multitone(void** device, multitone_profile* multitone_profile_o)	
功能描述	
初始化多音配置结构体，将结构体中的每个参数都赋初值。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **multitone_profile_o**	多音配置结构体。详见multitone_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.9 mod_init_stepsweep

int mod_init_stepsweep(void** device, stepsweep_profile* stepsweep_profile_o)	
功能描述	
初始化步进扫描配置结构体，将结构体中的每个参数都赋初值	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **stepsweep_profile_o**	步进扫描配置结构体。详见stepsweep_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.10 mod_init_rampsweep

int mod_init_rampsweep(void** device, rampsweep_profile* rampsweep_profile_o)	
功能描述	
初始化斜坡扫描配置结构体，将结构体中的每个参数都赋初值	
兼容性	2.0.18及之后版本支持

参数说明	
[in] **device**	设备句柄。
[out] **rampsweep_profile_o**	斜坡扫描配置结构体。详见rampsweep_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.11 mod_init_ofdm

```
int mod_init_ofdm (
    void** device,
    ofdm_profile* ofdm_profile_o
)
```

功能描述	
初始化正交频分复用配置结构体，将结构体中的每个参数都赋初值	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **ofdm_profile_o**	正交频分复用配置结构体。详见ofdm_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.12 mod_init_awgn

```
int mod_init_awgn (
    void** device,
    awgn_profile* awgn_profile_o
)
```

功能描述	
初始化高斯白噪声配置结构体，将结构体中的每个参数都赋初值	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **awgn_profile_o**	正交频分复用配置结构体。详见awgn_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。

9.13 mod_config_digitalmod

int mod_config_digitalmod(void** device, const digitalmod_profile* digitalmod_profile_in digitalmod_profile* digitalmod_profile_o)	
功能描述	
初始化数字调制配置结构体，将结构体中的每个参数都赋初值。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **digitalmod_profile_in**	输入数字调制配置结构体。详见digitalmod_profile结构体定义。
[out] **digitalmod_profile_o**	输出数字调制配置结构体。详见digitalmod_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_open之后调用此函数。
示例	请参考mod_generate_stream()函数相关示例。

9.14 mod_config_am

int mod_config_am(void** device, const am_profile* am_profile_in am_profile* am_profile_o)	
功能描述	
初始化AM调制配置结构体，将结构体中的每个参数都赋初值。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **am_profile_in**	输入AM调制配置结构体。详见am_profile结构体定义。
[out] **am_profile_o**	输出AM调制配置结构体。详见am_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_am之后调用此函数。

9.15 mod_config_fm

int mod_config_fm(void** device, const fm_profile* fm_profile_in fm_profile* fm_profile_o)	
功能描述	
配置配置FM调制参数，若输入配置不合理，将回写输出配置结构体为合理参数	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **fm_profile_in**	输入FM调制配置结构体。详见fm_profile结构体定义。
[out] **fm_profile_o**	输出FM调制配置结构体。详见fm_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_fm之后调用此函数。

9.16 mod_config_dsss

int mod_config _dsss(void** device, const dsss_profile* dsss_profile_in dsss_profile* dsss_profile_o)	
功能描述	
初始化直接序列扩频配置结构体，将结构体中的每个参数都赋初值。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **dsss_profile_in**	输入直接序列扩频配置结构体。详见dsss_profile结构体定义。
[out] **dsss_profile_o**	输出直接序列扩频配置结构体。详见dsss_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_dsss之后调用此函数。

9.17 mod_config_pulse

int mod_config _pulse(void** device, const pulse_profile* pulse_profile_in pulse_profile* pulse_profile_o)	
功能描述	
初始化脉冲调制配置结构体，将结构体中的每个参数都赋初值。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **pulse_profile_in**	输入脉冲调制配置结构体。详见pulse_profile结构体定义。
[out] **pulse_profile_o**	输出脉冲调制配置结构体。详见pulse_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_pulse之后调用此函数。

9.18 mod_config_multitone

int mod_config_multitone(void** device, const multitone_profile* multitone_profile_in multitone_profile* multitone_profile_o)	
功能描述	
初始化多音配置结构体，将结构体中的每个参数都赋初值。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **multitone_profile_in**	输入多音配置结构体。详见multitone_profile结构体定义。
[out] **multitone_profile_o**	输出多音配置结构体。详见multitone_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_multitone之后调用此函数。

9.19 mod_config_stepsweep

int mod_config_stepsweep(void** device, const stepsweep_profile* stepsweep_profile_in stepsweep_profile* stepsweep_profile_o)	
功能描述	
配置步进扫描参数，若输入配置不合理，将回写输出配置结构体为合理参数	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **stepsweep_profile_in**	输入步进扫描配置结构体。详见stepsweep_profile结构体定义。
[out] **stepsweep_profile_o**	输入步进扫描配置结构体。详见stepsweep_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_stepsweep之后调用此函数。

9.20 mod_config_rampsweep

int mod_config_rampsweep(void** device, const rampsweep_profile* rampsweep_profile_in rampsweep_profile* rampsweep_profile_o)	
功能描述	
配置斜坡扫描参数，若输入配置不合理，将回写输出配置结构体为合理参数	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[out] **rampsweep_profile_o**	斜坡扫描配置结构体。详见rampsweep_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_rampsweep之后调用此函数。

9.21 mod_config_ofdm

int mod_config_ofdm (void** device, const ofdm_profile* ofdm_profile_in ofdm_profile* ofdm_profile_o)	
功能描述	
配置正交频分复用参数，若输入配置不合理，将回写输出配置结构体为合理参数。	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **ofdm_profile_o**	输入正交频分复用配置结构体。详见ofdm_profile结构体定义。
[out] **ofdm_profile_o**	输出正交频分复用配置结构体。详见ofdm_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_ofdm之后调用此函数。

9.22 mod_config_awgn

int mod_config_awgn (void** device, const awgn_profile* awgn_profile_in awgn_profile* awgn_profile_o)	
功能描述	
初始化高斯白噪声配置结构体，将结构体中的每个参数都赋初值	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **awgn_profile_in**	输入高斯白噪声配置结构体。详见awgn_profile结构体定义。
[out] **awgn_profile_o**	输出高斯白噪声配置结构体。详见awgn_profile结构体定义。
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在成功执行mod_init_awgn之后调用此函数。

9.23 mod_generate_stream

int mod_generate_stream (**void** device,** **int32_t symbol_length,** **int16_t** iq_o,** **int32_t* length_o,** **double* sample_rate_o** **)**	
功能描述	
打开调制功能后，调用初始化函数赋初值，然后通过配置结构体函数下发参数，生成信号数据	
兼容性	2.0.18及之后版本支持
参数说明	
[in] **device**	设备句柄。
[in] **symbol_length**	符号长度，用于数字调制、直接序列扩频调制。
[out] **iq_o**	输出信号数据
[out] **length_o**	输出数据长度
[out] **sample_rate_o**	当前生成数据所用的采样率
返回值	0：成功；非0：错误代码，详见附录1。
调用约束	必须在执行 device_open_* 初始化之后调用此函数。
示例	

```
/* 启动配置与设备信息 */
int status = 0;
void* device = nullptr;
device_info dinfo;
channel ch[MAXCHANNELS];
uint16_t chn = 0;
uint8_t dnum = 0;
char version[50] = { "" };
int16_t* iq = nullptr;
int32_t length = 0;
double sampleRate = 0;
/* 设备启动 */
status = device_open_usb(&device, dnum, ch, &chn, &dinfo);
if (status != 0) {
   std::cout << "[ERROR] device_open_usb failed! status = " << status << std::endl;
   return status; // 若open失败  不进入后续的生成调制信号环节
```

```
}
/* 打开调制 */
status = mod_open(&device, version);
digitalmod_profile digitalModProfile_in;
digitalmod_profile digitalModProfile_out;
mod_init_digitalmod(&device, &digitalModProfile_in);

/* 配置参数 */
digitalModProfile_in.digitalmod_type = DigitalModType_BPSK;
digitalModProfile_in.symbolrate = 5e6;
digitalModProfile_in.sps = 4;
status = mod_config_digitalmod(&device, &digitalModProfile_in, &digitalModProfile_out);

/* 获取调制波形数据 */
uint32_t symbol_length = 1U << digitalModProfile_out.pn;
status = mod_generate_stream(&device, symbol_length, &iq, &length, &sampleRate);

// 延拓IQ，I和Q各超过65536个点
std::vector<int16_t> vector;
if (length < 65536 * 2) {
   int count = std::ceil(65536.0 * 2 / length);
   vector.resize(count * length);
   for (int i = 0; i < count; i++) {
      memcpy(vector.data() + i * length, iq, length * sizeof(int16_t));
   }
   length = count * length;
}

/* 配置射频 */
double fc = 1.0e9; // 输出频率 Hz
float level = 0.0f; // 输出功率 dBm
status = tx_config_ffm(ch, fc, level); // 配置为固定频点模式
print_status(status);

/* 配置触发 */
```

```
tx_trigger trg;
trg.source = TRIGGER_SOURCE_BUS;
status = channel_config_trigger(ch, &trg);
print_status(status);

/* 下发波形 */
uint16_t waveid = 0;
status = tx_clear_waveform(&device);
int16_t* final_waveform = vector.empty() ? iq : vector.data();
status = tx_download_waveform(&device, final_waveform, length/2, &waveid);
print_status(status);

/* 配置基带 */
int16_t waveforms = 1;                          // 播放波形的个数
const int16_t waveform[] = { waveid }; // 指定哪些序号的波形需要被播放
const int32_t repeat[] = { -1 };                // 每个播放波形的重复次数，-1表示无限重复
const double srate[] = { sampleRate }; // 每个波形的采样率
status = tx_config_playback(ch, waveform, repeat, srate, waveforms); //设置播放playback
print_status(status);
status = tx_config_output(ch, STATE_ON, STATE_ON); //配置输出
status = channel_start(ch); //启动通道
status = channel_trigger_bus(ch);
std::this_thread::sleep_for(std::chrono::seconds(10));
status = mod_close(&device); //关闭调制
if (status) {
   std::cout << __LINE__ << ", mod_close_status = " << status << std::endl;
}
status = device_close(&device); //关闭设备
return 0;
```

10. 结构体变量

10.1 device_info

device_info 详细定义	
uint16_t model	设备型号。
uint64_t uid_l64	低 64bit 设备序列号。
uint32_t uid_h32	高 32bit 设备序列号。
uint16_t hardware_version	硬件版本。
uint32_t mfw_version	MCU 固件版本。
uint32_t ffw_version	FPGA 固件版本。
uint32_t pmu_version	PMU 固件版本。
uint32_t agu_version	AGU 固件版本。
uint32_t bus_version	总线外设固件版本。
uint32_t eio_version	EIO 固件版本。
uint32_t bus_bandwidth	总线带宽，单位 MB/s。

10.2 supply_info

supply_info 详细定义	
float rf_voltage	电源端口电压，单位 V。
float rf_current	电源端口电流，单位 A。
float usb_voltage	USB 端口电压，单位 V。
float usb_current	USB 端口电流，单位 A。

10.3 eth_setting

eth_setting 详细定义	
eth_interface_type eth_interface	物理接口类型，详见 eth_interface_type 枚举的定义。
uint8_t eth_is_ipv6	ETH 协议类型（0：IPv4，1：IPv6）。
uint8_t eth_ip_address[16]	ETH 接口 IP 地址。
uint16_t eth_remote_port	ETH 接口远端端口。
int32_t eth_read_timeout	ETH 接口读取超时，单位 ms。

10.4 gnss_setting

gnss_setting 详细定义	
uint8_t antenna	天线路径选择，0 表示内部天线，1 表示外部天线。
uint8_t pps_en	PPS 使能状态。
float pps_x	pps 频率，单位 Hz。
float pps_delay	pps 延时，单位秒。

10.5 gnss_info

gnss_info 详细定义	
state locked	GNSS 锁定状态。0：未锁定；1：锁定。详见 state 枚举的定义。
uint8_t sat_nums	当前 GNSS 定位卫星数量。
uint8_t snr_max	当前 GNSS 定位卫星的 SNR 最大值。
uint8_t snr_avg	当前 GNSS 定位卫星的 SNR 平均值。
uint8_t snr_min	当前 GNSS 定位卫星的 SNR 最小值。
float latitude	纬度，北纬为正数，南纬为负数。
float longitude	经度，东经为正数，西经为负数。
float altitude	海拔，单位：米。
uint64_t ns_sinceepoch	ns 级系统时间戳。

10.6 channel

channel 详细定义	
void** device	通道所属设备句柄。
uint16_t num	通道号。
channel_type type	通道类型。详见 channel_type 枚举的定义。

10.7 tx_trigger

tx_trigger 详细定义	
trigger_source source	触发源。详见 trigger_source 枚举的定义。
trigger_edge edge	触发边沿，详见 trigger_edge 枚举的定义。总线触发下，该参数无效。
trigger_action action	触发动作；固定频点模式下，该参数无效。详见 trigger_action 枚举的定义。

int32_t count	触发响应次数；-1 表示无限次；>0 表示有限次响应，响应 count 次后停止，需重新触发。 固定频点模式下，该参数无效； 当触发动作为 TRIGGER_ACTION_HOP 时，表示一次触发切换 count 个射频状态，间隔时间为设置的驻留时间； 当触发动作为 TRIGGER_ACTION_SWEEP 时，表示一次触发进行 count 次扫描，每次扫描所包含的时间间隔为设置的驻留时间。

10.8 trigger_out

trigger_out 详细定义	
state enable	触发输出状态。详见 state 枚举的定义。
trigger_action action	触发动作。完成一次 hop 或扫描后，输出一个触发。详见 trigger_action 枚举的定义。
trigger_edge edge	触发边沿。详见 trigger_edge 枚举的定义。

10.9 digitalmod_profile

digitalmod_profile 详细定义	
mod_digitalmod_type digitalmod_type	触发输出状态。详见 mod_digitalmod_type 枚举的定义。
double symbolrate	符号速率，单位 Hz， 范围：最小采样率/每个符号样点数 ~ 最大采样率/每个符号样点数
double fskdeviation	FSK 频偏，单位 Hz,最大为 15 * symbolrate
int32_t pn	伪随机序列，4 ~ 32，建议最大不超过 24
mod_filter_type filter_type	滤波器类型，详见 mod_filter_type 枚举的定义
int32_t filter_symbols	滤波器符号数，偶数，2 ~ sps*filter_symbols <= 400, 默认取 8
int32_t sps	每个符号样点数，4 ~ 32 的偶数，默认取 4
double alpha	升余弦/根升余弦：0.025 ~ 1，高斯滤波：0.15 ~ 2.5，alpha 只对升余弦、根升余弦、高斯滤波有效

10.10 am_profile

am_profile 详细定义	
double rate	调制率，单位 Hz
double depth	调制深度，单位%
mod_shape_type shape	调制波形，详见 mod_shape_type 枚举的定义

10.11 fm_profile

fm_profile 详细定义	
double rate	调制率，单位 Hz
double deviation	调制频偏，单位 Hz
mod_shape_type shape	调制波形，详见 mod_shape_type 枚举的定义

10.12 dsss_profile

dsss_profile 详细定义	
double symbolrate	符号速率，单位 Hz
mod_digitalmod_type digitalmod_type	数字调制方式，详见 mod_digitalmod_type 枚举的定义
int32_t code	扩频码，取 PN 值，建议取 5
int32_t filter_symbols	滤波器符号数，偶数，2 ~ sps*filter_symbols <= 400, 默认取 8
int32_t sps	每个符号样点数，4 ~ 32 的偶数，默认取 4
double alpha	升余弦/根升余弦：0.025 ~ 1，高斯滤波：0.15 ~ 2.5，alpha 只对升余弦、根升余弦、高斯滤波有效
int32_t sequenceseed	伪随机序列初始值，1 ~ 2147483647
mod_filter_type filter_type	滤波器类型，详见 mod_filter_type 枚举的定义

10.13 pulse_profile

pulse_profile 详细定义	
double width	脉宽，单位 s
double period	周期，单位 s，周期一定大于脉宽

10.14 multitone_profile

multitone_profile 详细定义	
double tonephase	相位，[0，2*pi]
int32_t tonecount	单音数量
double freqspacing	频率间隔，单位 Hz

10.15 stepsweep_profile

stepsweep_profile 详细定义	
double center	中心频率，单位 Hz
double span	扫宽，单位 Hz
int32_t points	点数
double dwelltime	驻留时间，单位 s

10.16 rampsweep_profile

rampsweep_profile 详细定义	
double span	扫宽，单位 Hz
double sweeptime	扫描时间，单位 s
double period	周期，单位 s

10.17 ofdm_profile

ofdm_profile 详细定义	
double idleinterval	发射突发之间的空闲时间，单位 s
int32_t fftsize	FFT 个数，支持 16、32、64、128、256、512、1024、2048， 且 fftsize > guardbandcarriers_left + guardbandcarriers_right
int32_t symbolcount	符号数量
int32_t guardbandcarriers_left	左侧保护载波个数
int32_t guardbandcarriers_right	右侧保护载波个数
int32_t guardinterval	保护间隔，单位为%
int32_t window_length	窗长度，单位为%
uint8_t windowed	是否加窗

mod_digitalmod_type digitalmod_type	数字调制方式，注意，仅支持 DigitalModType_BPSK、DigitalModType_QPSK、DigitalModType_PSK8、DigitalModType_PSK16、DigitalModType_QAM16、DigitalModType_QAM64、DigitalModType_QAM256
uint8_t nulldc	直流是否置空
double samplerate	采样率 195.3125e3 ~ 125e6，默认 20MHz

10.18 awgn_profile

awgn_profile 详细定义	
double bandwidth	发射突发之间的空闲时间，单位 s
int32_t length	FFT 个数，支持 16、32、64、128、256、512、1024、2048，且 fftsize > guardbandcarriers_left + guardbandcarriers_right

11. 枚举变量

11.1 state

STATE_OFF	全局配置，0；
STATE_ON	全局配置，1。

11.2 fan_mode

FAN_ON	风扇常开；
FAN_OFF	风扇常关；
FAN_AUTO	根据温度自动调整风扇状态。

11.3 eth_interface_type

ETH_HIGHSPEED	高速以太网总线接口；
ETH_STANDARD	普通以太网总线接口。

11.4 referenceclock_source

REFCLKSOURCE_INTERNAL	内部参考时钟源；
REFCLKSOURCE_EXTERNAL	外部参考时钟源。

11.5 lomode

LOMODE_AUTO	在该模式下，仪器根据当前输出频率、频率步进及工作状态自动选择最优工作参数；
LOMODE_LOWSPUR	该模式优化本振的相位噪声性能，通过调整锁相环控制策略，提高信号的相位稳定性；
LOMODE_LOWPHASENOISE	该模式针对本振输出频谱进行优化，有效抑制分数杂散及相关寄生分量，提高频谱纯净度。

11.6 channel_type

CHANNELTYPE_RX	接收通道；
CHANNELTYPE_TX	发射通道；
CHANNELTYPE_VX	矢量网络分析通道。

11.7 trigger_action

TRIGGER_ACTION_HOP	执行一次跳频；
TRIGGER_ACTION_SWEEP	执行一次完整扫描。

11.8 trigger_edge

TRIGGER_EDGE_RISING	上升沿触发；
TRIGGER_EDGE_FALLING	下降沿触发。

11.9 trigger_source

TRIGGER_SOURCE_BUS	总线触发；在该触发下，触发边沿无效；
TRIGGER_SOURCE_EXTERNAL	外部触发；
TRIGGER_SOURCE_XPPS	GNSS 秒脉冲触发；
TRIGGER_SOURCE_LEVEL	电平触发，仅用于 Rx；
TRIGGER_SOURCE_FREQMASK	频谱模版触发，仅用于 Rx。

11.10 mod_digitalmod_type

DigitalModType_BPSK	BPSK
DigitalModType_DBPSK	DBPSK
DigitalModType_QPSK	QPSK
DigitalModType_DQPSK	DQPSK
DigitalModType_OQPSK	OQPSK
DigitalModType_Pi4DQPSK	Pi4DPSK
DigitalModType_PSK8	8PSK
DigitalModType_D8PSK	D8PSK
DigitalModType_PSK16	16PSK
DigitalModType_QAM16	16QAM
DigitalModType_QAM64	64QAM
DigitalModType_QAM256	256QAM
DigitalModType_QAM1024	1024QAM
DigitalModType_ASK2	2ASK
DigitalModType_FSK2	2FSK
DigitalModType_FSK4	4FSK
DigitalModType_FSK8	8FSK
DigitalModType_FSK16	16FSK
DigitalModType_ASK4	4ASK
DigitalModType_ASK8	8ASK
DigitalModType_APSK16	16ASK
DigitalModType_QAM8	8QAM

11.11 mod_filter_type

FilterType_Rectangular	矩形滤波器
FilterType_RaisedCosine	升余弦滤波器
FilterType_RootRaisedCosine	根升余弦滤波器
FilterType_Gaussian	高斯滤波器
FilterType_HalfSine	半正弦滤波器

11.12 mod_shape_type

ModShape_Sine	正弦波
ModShape_Square	方波
ModShape_Triangle	三角波
ModShape_Ramp	锯齿波

附录 Appendix 1： API 返回值索引

错误代码	错误/警告原因	类型[1]	处理
0	无错误	-	无需处理，可正常执行后续过程。
-1	总线打开错误	错误	检查设备供电、数据线连接，检查驱动程序是否正确安装。排除错误后需要重新调用 device_open_usb 打开设备。
-8	总线传输错误	错误	重新插拔设备，若仍无法解决，请联系官方技术支持。
-29	发射功率校准文件丢失[2]	错误	检查发射功率校准文件是否放置在规定的目录下。排除错误后需要重新调用 device_open_usb 打开设备。
-43	IQ 校准文件丢失[2]	错误	检查 IQ 校准文件是否放置在规定的目录下。排除错误后需要重新调用 device_open_usb 打开设备。
10	数据传输超时	警告	等待 10ms 后，重新尝试下发。
42	使用默认发射功率校准文件	警告	不影响设备继续运行，但射频输出功率存在偏差。
44	使用默认 IQ 校准文件	警告	不影响设备继续运行，但精度会存在偏差。
98	参数超界发生钳位	警告	将越界参数强制修改回设备参数范围，可通过 query 查询当时的真实值，不影响设备继续运行。
99	发射功率过低，但未越界	警告	功率可能不准，不影响设备继续运行，请联系官方技术支持。
100	发射功率过高，但未越界	警告	功率可能不准，不影响设备继续运行，请联系官方技术支持。
15	系统时钟失锁	警告	测试结果可能不准，不影响设备继续运行，请联系官方技术支持。
16	ADC 时钟失锁	警告	测试结果可能不准，不影响设备继续运行，请联系官方技术支持。
18	射频本振失锁	警告	测试结果可能不准，不影响设备继续运行，请联系官方技术支持。
-601	调制许可证文件缺失	错误	请将 xx._mod.lic 文件放到/CalFile/文件夹中。
-602	调制许可文件验证失败	错误	通常为许可证内容错误，请联系官方技术支持。
-603	设备校验失败	错误	创建设备对象时，设备校验失败，请检查设备是否正常打开，并联系官方技术支持。
-604	对象创建失败	错误	创建设备对象时，对象创建失败，请检查设备是否正常打开，并联系官方技术支持。
-605	对象释放失败	错误	释放设备对象失败，请联系官方技术支持。
-606	调制类型错误	错误	生成信号时，输入调制类型错误，请修改调制类型并下发，重新生成信号。

-607	调制参数错误	错误	生成信号时，输入调制参数错误，请修改调制参数并下发，重新生成信号。
-608	缺少调制库	错误	h2_api 库的同路径下没有调制库，请将 genSignalWave.dll(Windwos)或 libgenSignalWave.so(Linux)放到 h2_api 库的同路径下。
-609	参数无法配置	错误	当前参数无法配置，已配置上一次可用参数。

[1] 类型为“错误”时，需要立即排除问题，并重新打开设备，否则设备无法继续运行后续流程。类型为“警告”时，设备可继续后续流程，无需关闭或重新打开设备，但仍然建议针对具体的返回值与当前的应用场景选择性的处理。

[2] 对于返回值为-29、-43 的情况，还需要确认文件存放路径是否为全英文路径。若路径中包含中文字符，调用 API 时也会提示文件加载失败。

www.harogic.cn
cninfo@harogic.com
+025-8330 5049