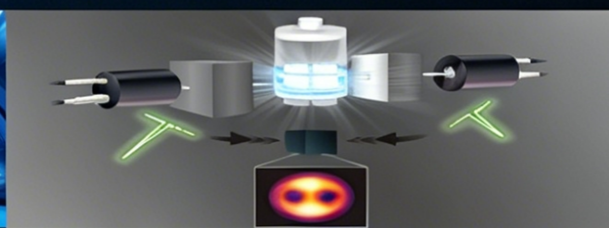
Venus Lite

Time-Digital Converter
Time Tagger

- High time measurement resolution: 0.975 ps
- Excellent timing performance: ~ 3 ps RMS jitter
- Low time measurement dead time: ~ 2 ns
- 16/8/4 channels per device
- Online coincidence measurement/frequency analysis/TOT/TCSPC
- Support g^2 curve/IEEE 1139 clock analysis
- Maximum Count Rate: 250 M cps/ch
- Maximum Clock Analysis Frequency: 500 MHz
- USB 3.0/1 GbE/10 GbE/40 Gbps QSFP communication interface
- Fully open and customizable C/C++/Python/MATLAB/LabVIEW API/SDK
- Data acquisition/analysis host software (Windows/Linux)
- Scalable up to 256 devices (4096 Ch) for synchronized measurement

Quantum Communication and Calculation

Coincidence Measurement



Fluorescence Measurement

Frequency Analysis

Single Photon Detection

Venus lite series.

16/8 Channel, 0.975 ps

Streaming Time-Digital Converter

Venus Lite API User Manual

UG-01-3-050-4, Jun. 2026

Chronos Technology specializes in the development of ultra-high precision measurement equipment, offering nanosecond-level synchronization solutions tailored for the quantum, medical, and industrial sectors. For more product information, please visit our official website- *www.chronosci.com*

1. 开发者 API 接口

9.1 C API

9.1.1 概述

CINST API 是用于和高精度测量设备（比如：TDC 时间数字转换器设备）进行交互操作的 C API，提供设备连接、配置、数据采集和分析等功能。

这份文档涵盖了 CINST API 的主要功能和使用方法，可用于开发基于该接口的应用程序。

9.1.2 初始化和销毁

注意事项：

- config_path 配置文件可以是绝对路径，也可以是相对路径。如果是相对路径，先查找系统环境变量 CSP_BASE_PATH 指定的目录；然后查找下面 base_dir 指定的基础目录，最后会在程序根目录查找；
- base_dir 可以指定基础目录路径，其他相对路径都可以基于这个基础路径。

【API 定义】

API 函数	参数	返回值	功能描述	备注
ci_create(const char* config_path, const char* base_dir)	config_path：配置文件路径 base_dir:基础目录路径	ci_handle_t	创建实例	
ci_destroy(ci_handle_t handle)	handle：对象句柄（ci_handle_t）	-	销毁实例	
ci_cleanup()	-	-	销毁所有实例	

9.1.3 设备连接

注意事项：

- 可以先调用 get_devices 获取设备后，再进行连接；
- 操作设备前，必须先连接设备；
- 操作结束后，不再操作设备，请先断开设备连接。

【设备接口类型定义-cinst_interface_type_t】

定义	值	说明
CINST_USB3_DEV	1	Usb3.0 接口
CINST_UDP_DEV	2	千兆网网口
CINST_UDP_W_DEV	3	万兆网网口

【API 定义】

API 函数	参数	返回值	功能描述	说明
ci_get_devices（ci_handle_t handle, cinst_device_info_t* out_array, int32_t capacity）	handle：对象句柄 out_array：设备结构体数组 capacity：数组大小	int-设备数量	获取可用设备列表	设备信息包含类型和名称 支持两段式获取，第一次传入 NULL 数值，返回数组大小；第二次返回实际数组值
ci_get_device_num(ci_handle_t handle)	handle：对象句柄	int-设备数量	获取可用设备数量	
ci_get_device（ci_handle_t handle, int32_t index, int32_t* out_type, char* out_name_utf8, int32_t name_size）	handle：对象句柄 Index：设备 index out_type:设备接口类型 out_name_utf8：设备接口名称 name_size：名称缓存大小	int - 状态码	获取指定设备的接口类型和名称	
ci_connect(ci_handle_t handle, int32_t	handle：对象句柄 device_type：设备	int - 状态码	连接指定类型设备	具体类型定义请详见 ConnType

device_type)	接口类型			定义。
disconnect(ci_handle_t handle)	handle：对象句柄	int - 状态码	断开当前设备连接	

9.1.4 错误处理

【错误码定义-cinst_error_t】

定义	值	说明
CERROR_CONFIG_NOT_FOUND	2000	配置文件不存在
CERROR_INST_DEVICE_NOT_FOUND	2001	设备未找到
CERROR_INST_DEVICE_NOT_CONNECTED	2002	设备未连接
CERROR_INST_DEVICE_ALREADY_RUNNING	2003	设置已运行
CERROR_INST_START_PROTOCOL_FRAMEWORK_FAILED	2004	启动协议层失败
CERROR_INST_PROTOCOL_FRAMEWORK_NOT_RUNNING	2005	协议层未运行
CERROR_INST_PROTOCOL_FRAMEWORK_INTERNAL	2006	协议层内部错误
CERROR_INST_INVALID_ACQ_TYPE	2007	错误的数据采集方式
CERROR_INST_ACQ_DATA_FAILED	2008	数据采集失败
CERROR_INST_PROCESS_ANALYSIS_FAILED	2009	数据分析失败
CERROR_INST_DATA_PROCESS_NOT_STOPPED	2010	上一次数据分析任务未完成
CERROR_INST_INVALID_CALIBRATION_PARAM	2011	错误的标定参数
CERROR_INST_CALIBRATION_FAILED	2012	设备标定失败
CERROR_INST_CONVERT_BINFILE_FAILED	2013	转换 BIN 文件失败
CERROR_INST_INVALID_PARAMETER	2014	错误的参数
CERROR_INST_TERMINATION_CONFIG_UNSUPPORTED	2015	不支持阻抗参数配置
CERROR_INST_ENABLE_CHANNEL_FAILED	2016	通道使能失败
CERROR_INST_DISABLE_CHANNEL_FAILED	2017	通道禁能失败
CERROR_INST_SET_CHANNELS_ENABLED_FAILED	2018	设置所有通道是否使能失败

【API 定义】

API 函数	参数	返回值	功能描述	备注

ci_set_error_callback(ci_handle_t handle, ci_error_cb_t cb, void* user_data)	handle：对象句柄 cb：错误回调函数 user_data:用户自定义数据	-	注册错误回调	
ci_get_last_error(ci_handle_t handle)	handle：对象句柄	int - 错误码	获取最近的错误码	
ci_get_last_error_message(ci_handle_t handle)	handle：对象句柄	const char* - 错误信息	获取最近的错误信息	
ci_info(ci_handle_t handle, const char* message)	handle：对象句柄 message：日志信息	-	记录 INFO 级别日志	
ci_warn(ci_handle_t handle, const char* message)	handle：对象句柄 message：日志信息	-	记录 WARNING 级别日志	
ci_error(ci_handle_t handle, const char* message)	handle：对象句柄 message：日志信息	-	记录 ERROR 级别日志	
ci_critical(ci_handle_t handle, const char* message)	handle：对象句柄 message：日志信息	-	记录 CRITICAL 级别日志	

错误回调函数定义：

typedef void (*ci_error_cb_t)(int32_t error_code, const char* description_utf8, void* user_data);

9.1.5 设备自检

设备使用前，可以先进行设备自检操作，设置各通道和全局配置的初始默认值。设备自检后，在 selfcheck 目录下会生成自检日志文档，可以查看自检状态。

【API 定义】

API 函数	参数	返回值	功能描述	备注
ci_self_check(ci_handle_t handle)	handle：对象句柄	int - 状态码	设备自检	

9.1.6 通道配置

注意事项：

- 参考通道不可设置 DAC、Termination、Hysteresis 和 A/D Integration Point 值；
- A/D Integration Point 值，仅对 A/D 通道有效（设备的奇数通道）；
- Termination 配置，仅对部分型号支持（Venus Ultra 系列版本）。

【阻抗类型定义-cinst_term_type_t】

定义	值	说明
CINST_TERM_50	0	阻抗 50 Ohm（默认值）
CINST_TERM_1M	1	阻抗 1M Ohm

【迟滞电压类型定义-cinst_hts_type_t】

定义	值	说明
CINST_HTS_30_MV	0	迟滞电压 30mV（默认值）
CINST_HTS_40_MV	1	迟滞电压 40mV
CINST_HTS_70_MV	2	迟滞电压 70mV
CINST_HTS_1_MV	3	迟滞电压 1mV

【边沿触发类型定义-cinst_edge_type_t】

定义	值	说明
CINST_RISING_EDGE	0	上升沿
CINST_FALLING_EDGE	1	下降沿
CINST_PMIDDLE	2	正中间

CINST_NMIDDLE	3	负中间

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
ci_set_dac(ci_handle_t handle, int32_t channel, double value)	handle：对象句柄 channel：通道号 value: DAC 值	int - 状态码	设置通道 DAC 电压	-1500~1500mV
ci_set_termination(ci_handle_t handle, int32_t channel, int32_t value)	handle：对象句柄 channel：通道号 value：阻抗类型值	int - 状态码	设置通道阻抗类型	详见 cinst_term_type_t 定义
ci_set_hysteresis(ci_handle_t handle, int32_t channel, int32_t value)	handle：对象句柄 channel：通道号 value：迟滞电压类型值	int - 状态码	设置通道迟滞电压类型	详见 cinst_hts_type_t 定义
ci_set_inter_delay(ci_handle_t handle, int32_t channel, int32_t value)	handle：对象句柄 channel：通道号 value：延迟时间值	int - 状态码	设置通道时间延迟值	范围：0~1000000ps
ci_set_dead_time(ci_handle_t handle, int32_t channel, int32_t value)	handle：对象句柄 channel：通道号 value：死时间	int - 状态码	设置通道死时间	范围：2~130000ps
ci_set_edge_type(ci_handle_t handle, int32_t channel, int32_t type)	handle：对象句柄 channel：通道号 value：边沿触发类型	int - 状态码	设置边沿触发类型	详见 cinst_edge_type_t 定义
ci_set_ad_integration_point(ci_handle_t handle, int32_t channel, int32_t value)	handle：对象句柄 channel：通道号 value：积分点数	int - 状态码	设置 A/D 面积积分值	范围：0~65535
ci_enable_channel(ci_handle_t handle, int32_t channel)	handle：对象句柄 channel：通道号	int - 状态码	设置通道使能	
ci_disable_channel(ci_handle_t handle, int32_t channel)	handle：对象句柄 channel：通道号	int - 状态码	设置通道禁能	
ci_enable_all_channels(ci_handle_t handle)	handle：对象句柄	int - 状态码	设置所有通道使能	

ci_disable_all_channels(ci_handle_t handle)	handle：对象句柄	int - 状态码	设置所有通道禁能	
ci_get_dac(ci_handle_t handle, int32_t channel, double* out_value)	handle：对象句柄 channel：通道号 out_value:返回值指针	int - 状态码	获取通道 DAC 值	
ci_get_termination(ci_handle_t handle, int32_t channel, int32_t* out_value)	handle：对象句柄 channel：通道号 out_value:返回值指针	int - 状态码	获取通道阻抗类型值	
ci_get_hysteresis(ci_handle_t handle, int32_t channel, int32_t* out_value)	handle：对象句柄 channel：通道号 out_value:返回值指针	int - 状态码	获取通道迟滞电压类型值	
ci_get_inter_delay(ci_handle_t handle, int32_t channel, int32_t* out_value)	handle：对象句柄 channel：通道号 out_value:返回值指针	int - 状态码	获取时间延迟值	
ci_get_dead_time(ci_handle_t handle, int32_t channel, int32_t* out_value)	handle：对象句柄 channel：通道号 out_value:返回值指针	int - 状态码	获取通道死时间	
ci_get_edge_type(ci_handle_t handle, int32_t channel, int32_t* out_value)	handle：对象句柄 channel：通道号 out_value:返回值指针	int - 状态码	获取边沿触发类型	
ci_get_ad_integration_point(ci_handle_t handle, int32_t channel, int32_t* out_value)	handle：对象句柄 channel：通道号 out_value:返回值指针	int - 状态码	获取 A/D 面积积分值	
ci_is_channel_enabled(ci_handle_t handle, int32_t channel, int8_t* out_value)	handle：对象句柄 channel：通道号 out_value:返回值指针	int - 状态码	获取通道使能状态	

9.1.7 全局配置

注意事项：

- 测试模式类型主要用于仿真和测试，正常使用采用 RAW 模式；
- 切换 TDC 时钟来源会涉及到设备的重新校准，耗时较长（一般在 20s）；
- 系统复位操作会重置设备所有设置，耗时较长（一般在 10s 左右），请谨慎操作；
- 板卡 ID 设置主要用于多设备并联的场景，用于区分不同设备。

【TDC 时钟来源类型定义-cinst_clock_source_t】

定义	值	说明
CINST_INTERNAL_CLOCK	0	内部时钟
CINST_EXTERNAL_CLOCK	1	外部时钟

【数据模式定义-cinst_data_mode_t】

定义	值	说明
CINST_DATA_MODE_TIMESTAMP	0	时间模式
CINST_DATA_MODE_TIME ZONE	1	时区模式
CINST_DATA_MODE_AD	2	A/D 模式
CINST_DATA_MODE_AREA	3	面积测量模式
CINST_DATA_MODE_REF_COINS	4	参考符合模式
CINST_DATA_MODE_GLOBAL_COINS	5	全局符合模式
CINST_DATA_MODE_TOT	6	过阈时间测量模式
CINST_DATA_MODE_AREA_COINS	7	能谱-全局符合模式
CINST_DATA_MODE_DUAL_TIMESTAMP	8	双沿时间模式
CINST_DATA_MODE_PERIOD	9	周期测量模式
CINST_DATA_MODE_SINGLE_COINS	11	单符合模式
CINST_DATA_MODE_GLOBAL_COINS_TIMESTAMP	12	全局符合（带时间戳）模式

【测试模式定义-cinst_test_mode_t】

定义	值	说明
CINST_TEST_MODE_RAW	0	RAW 原始数据模式
CINST_TEST_MODE_ALL_0	1	全 0 模式
CINST_TEST_MODE_ALL_1	2	全 1 模式
CINST_TEST_MODE_TOGGLE	3	转换模式
CINST_TEST_MODE_INCREASE	4	递增模式
CINST_TEST_MODE_DECREASE	5	递减模式
CINST_TEST_MODE_TRIGGER_1M	6	1MHz 内部触发模式

【数据带宽类型定义-cinst_test_data_bandwidth_t】

定义	值	说明
CINST_TEST_DATA_BANDWIDTH_100M	0	100M 带宽
CINST_TEST_DATA_BANDWIDTH_1000M	1	1000M 带宽
CINST_TEST_DATA_BANDWIDTH_3000M	2	3000M 带宽
CINST_TEST_DATA_BANDWIDTH_6400M	3	6400M 带宽

【A/D 面积积分缩放类型定义-cinst_ad_area_scale_t】

定义	值	说明
CINST_AREA_SCALE_1	0	1 倍
CINST_AREA_SCALE_1_2	1	1/2 倍
CINST_AREA_SCALE_1_4	2	1/4 倍
CINST_AREA_SCALE_1_8	3	1/8 倍
CINST_AREA_SCALE_1_16	4	1/16 倍
CINST_AREA_SCALE_1_32	5	1/32 倍
CINST_AREA_SCALE_1_64	6	1/64 倍
CINST_AREA_SCALE_1_128	7	1/128 倍

【数据采集方式定义-cinst_acq_type_t】

定义	值	说明

CINST_ACQ_SOFTWARE	0	软件方式
CINST_ACQ_HARDWARE	1	硬件方式

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
ci_set_tdc_clock_source(ci_handle_t handle, int32_t cs_type)	handle：对象句柄 cs_type: 时钟源类型	int - 状态码	设置 TDC 时钟源	详见 cinst_clock_source_t 定义
ci_set_data_mode(ci_handle_t handle, int32_t data_mode)	handle：对象句柄 data_mode: 数据模式	int - 状态码	设置数据模式	详见 cinst_data_mode_t 定义
ci_set_test_mode(ci_handle_t handle, int32_t test_mode)	handle：对象句柄 test_mode: 测试模式	int - 状态码	设置测试模式	详见 cinst_test_mode_t 定义
ci_set_test_data_bandwidth(ci_handle_t handle, int32_t bandwidth_type);	handle：对象句柄 bandwidth_type: 带宽类型	int - 状态码	设置测试数据带宽	详见 cinst_test_data_bandwidth_t 定义
ci_set_board_id(ci_handle_t handle, int32_t board_id);	handle：对象句柄 board_id: 板卡 ID	int - 状态码	设置新板卡 ID	范围：0~255
ci_set_coins_time_window(ci_handle_t handle, int32_t coins_value)	handle：对象句柄 coins_value: 时间窗口	int - 状态码	设置符合时间窗值	范围: 0~16000000 ps
ci_set_ad_area_scale(ci_handle_t handle, int32_t scale_value)	handle：对象句柄 scale_value:	int - 状态码	设置 A/D 面积积分缩放类型	详见 cinst_ad_area_scale_t 定义

	缩放类型			
ci_set_acq_type(ci_handle_t handle, int32_t acq_type)	handle：对象句柄 acq_type: 数据采集方式	int - 状态码	设置数据采集方式	详见 cinst_acq_type_t 定义
ci_tdc_resetn(ci_handle_t handle)	handle：对象句柄	int - 状态码	TDC 复位	
ci_system_resetn(ci_handle_t handle)	handle：对象句柄	int - 状态码	系统复位	
ci_get_tdc_clock_source(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取 TDC 时钟源	
ci_get_data_mode(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取数据模式	
ci_get_test_mode(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取测试模式	
ci_get_test_data_bandwidth(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取测试数据带宽	
ci_get_board_id(ci_handle_t handle)	handle：对象句柄	int - 板卡 ID 号	获取板卡 ID	如果失败，会返回-1
ci_get_coins_time_window(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取符合时间窗值	
ci_get_ad_area_scale(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取 A/D 面积积分缩放类型	
ci_get_acq_type(ci_handle_	handle：对象句	int - 状态码	获取数据采集方	

t handle, int32_t* out_value)	柄 out_value:返回值指针		式	

9.1.8 状态查询

【设备状态结构体-cinst_device_status_t】

定义	类型	说明
temperature	int	设备核心温度
current_1	float	电路 1 电流（mA）
current_2	float	电路 2 电流（mA）
current_3	float	电路 3 电流（mA）
current_4	float	电路 4 电流（mA）
current_5	float	电路 5 电流（mA）
current_6	float	电路 6 电流（mA）
current_7	float	电路 7 电流（mA）
current_8	float	电路 8 电流（mA）

【API 定义】

API 函数	参数	返回值	功能描述	注意事项
ci_get_channel_num(ci_handle_t handle)	handle:对象句柄	int - 设备通道数	获取通道数量	获取设备通道数，不包含参考通道
ci_get_device_type(ci_handle_t handle)	handle:对象句柄	int - 设备类型值	获取设备类型	0x1：Venus(TDC) 0x2: Mercury(多道分析仪)
ci_get_device_mode(ci_handle_t handle)	handle:对象句柄	int - 设备型号值	获取设备型号	0x1: Venus Lite-32/Mercury-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4

				0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0x10 - Venus Lite 4 0x11 - Venus Lite 2
ci_get_product_sn(ci_handle_t handle, char* out_value, int32_t size)	handle: 对象句柄 out_value :返回值指针 size：字符串大小	int - 状态码	获取产品序列号	
ci_get_device_status(ci_handle_t handle, cinst_device_status_t* out_data)	handle: 对象句柄 out_data: 返回数据指针	int - 状态码	获取设备状态	包含温度和各电路电流，详见 cinst_device_status_t 定义
ci_get_device_temp(ci_handle_t handle, int32_t* out_temp)	handle: 对象句柄 out_temp: 返回设备核心温度	int - 状态码	获取设备核心温度	
ci_get_fan_speed(ci_handle_t handle)	handle: 对象句柄	int - 风扇转速	获取设备风扇转速（单位：RPM）	
ci_get_fw_version(ci_handle_t handle, char* out_value, int32_t size)	handle: 对象句柄 out_value :返回固件版本号字符串 size：字符串长度	int - 状态码	获取设备固件版本号	
ci_get_sw_version(ci_handle_t handle,	handle: 对象句柄	int - 状态码	获取设备软件版本号	

char* out_value, int32_t size)	out_value :返回软件版本号字符串 size：字符串长度			

9.1.9 网络配置

注意事项：

- 本机 IP 和设备 IP 需要在同一个网段；
- 注意本地端口的占用情况，请不要设置已被占用的端口号；
- 重新修改 IP 地址和网络端口后，请重连设备进行操作；
- 重新设置 MAC 地址后，需要重启网络。

【API 定义】

API 函数	参数	返回值	功能描述
ci_set_ips(ci_handle_t handle, const char* local_ip, const char* device_ip)	handle：对象句柄 local_ip：本地 IP device_ip：设备 IP	int - 状态码	设置千兆网口 IP 地址
ci_set_net_ports(ci_handle_t handle, int32_t local_port, int32_t device_port)	handle：对象句柄 local_port：本地端口 device_port：设备端口	int - 状态码	设置千兆网口端口号
ci_set_wips(ci_handle_t handle, const char* local_ip, const char* device_ip)	handle：对象句柄 local_ip：本地 IP device_ip：设备 IP	int - 状态码	设置万兆网口 IP 地址
ci_set_wnet_ports(ci_handle_t handle, int32_t local_port, int32_t device_port)	handle：对象句柄 local_port：本地端口 device_port：设备端口	int - 状态码	设置万兆网口端口号
ci_get_ips(ci_handle_t handle, char* out_local_ip, int32_t local_ip_size, char* out_device_ip, int32_t device_ip_size)	handle：对象句柄 out_local_ip:返回 Local IP 指针 local_ip_size:字符数组大小	int - 状态码	获取千兆网口 IP 地址

	out_device_ip:返回的 Device IP 指针 device_ip_size:字符数组大小		
ci_get_net_ports(ci_handle_t handle, int32_t* out_local_port, int32_t* out_device_port)	handle：对象句柄 out_local_port:返回 Local Port 指针 out_device_port:返回 Device Port 指针	int - 状态码	获取千兆网口端口号
ci_get_wips(ci_handle_t handle, char* out_local_ip, int32_t local_ip_size, char* out_device_ip, int32_t device_ip_size)	handle：对象句柄 out_local_ip:返回 Local IP 指针 local_ip_size:字符数组大小 out_device_ip:返回的 Device IP 指针 device_ip_size:字符数组大小	int - 状态码	获取万兆网口 IP 地址
ci_get_wnet_ports(ci_handle_t handle, int32_t* out_local_port, int32_t* out_device_port)	handle：对象句柄 out_local_port:返回 Local Port 指针 out_device_port:返回 Device Port 指针	int - 状态码	获取万兆网口端口号
ci_set_mac(ci_handle_t handle, const char* mac)	handle：对象句柄 mac:网口 MAC 地址	int - 状态码	设置千兆网口 MAC 地址
ci_get_mac(ci_handle_t handle, char* out_mac, int32_t size)	handle：对象句柄 out_mac:返回 mac 地址指针 size:字符数组大小	int - 状态码	获取千兆网口 MAC 地址
ci_set_wmac(ci_handle_t handle, const char* mac)	handle：对象句柄 mac:网口 MAC 地址	int - 状态码	设置万兆网口 MAC 地址
ci_get_wmac(ci_handle_t handle, char* out_mac, int32_t size)	handle：对象句柄 out_mac:返回 mac 地址指针	int - 状态码	获取万兆网口 MAC 地址

	size:字符数组大小		

9.1.10 数据采集

注意事项：

- 数据采集 API 是异步的，API 调用后，不会阻塞线程，会立即返回，需要程序自己处理等待逻辑；
- API 返回失败，可以在错误回调中接收到具体的错误信息，或调用 get_last_error 方法查看错误代码和错误信息；
- 提供了实时获取当前数据的接口，可以读取数据到内存进行实时处理，但必须要提前调用设置数据缓存接口；
- 可以调用 Stop API 去强制停止当前的数据采集任务；如果不调用，采集时间到了后，系统会自动调用 Stop API，停止数据采集；
- 启用十六进制格式输出，写文件速率较慢（默认二进制数据格式，文件写入更高效），耗时较长，请注意数据文件大小。

【API 定义】

API 函数	参数	返回值	说明
ci_enable_hex_dataformat(ci_handle_t handle)	handle：对象句柄	int - 状态码	启用十六进制数据格式
ci_disable_hex_dataformat(ci_handle_t handle)	handle：对象句柄	int - 状态码	禁用十六进制数据格式
ci_is_hex_dataformat(ci_handle_t handle, int8_t* out_value)	handle：对象句柄 out_value:返回布尔值	int - 状态码	是否启用十六进制数据格式
ci_set_rawdata_cache(ci_handle_t handle, int buffer_size_mb)	handle：对象句柄 buffer_size_mb:数据缓存大小（单位：MB）	int - 状态码	设置 RAW 数据缓存大小，支持实时读取 单位：MB
ci_acq_rawdata(ci_handle_t handle, int32_t data_mode, int32_t duration, const char*	handle：对象句柄 data_mode：数据模式 duration：持续时	int - 状态码	开始进行数据采集（异步模式） <data_mode>请详见前面的

folder_path, const char* file_name, int32_t max_volumesize_mb, int32_t force_close)	间 file_path：数据文件目录（不设置，默认是 raw 目录） file_name：数据文件名 max_volumesize_mb：文件分卷大小 force_close：是否强制停止文件保存(1-强制停止；0-不强制停止，待内存数据都保存到文件)		DataMode 定义 <duration> 采集时间：单位-秒 <file_name> 未明确后缀名情况下，如果启用十六进制格式输出，后缀名为“.dat”;如果未启用，默认是二进制格式输出，后缀名为“.bin” <max_volumesize_mb> 如果分卷大小小于等于 0，文件不分卷
ci_fetch_rawdata(ci_handle_t handle, int32_t* out_size, int32_t timeout_ms)	handle：对象句柄 out_size:返回数组大小 timeout_ms：超时时间（单位：ms）	char*	实时读取 Raw 数据 返回 char 数组
ci_stop_acq(ci_handle_t handle)	handle：对象句柄	int - 状态码	停止数据采集
ci_convert_to_hex(ci_handle_t handle, const char* bin_file, const char* hex_file)	handle：对象句柄 bin_file：二进制原始数据文件路径 hex_file:输出的十六进制文件路径	int - 状态码	转换 BIN 文件为 HEX 文件 如果 hex_file 输入为空，默认生成的十六进制文件在二进制文件同级目录下

9.1.11 数据分析

注意事项：

- 数据分析 API 是异步的，API 调用后，不会阻塞线程，会立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或调用 ci_get_last_error/ci_get_last_error_message 方法查看错误代码和错误信息；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到了后，系统会自动调用 Stop API，停止分析。

【Start-stop 分析通道分组结构体-cinst_chan_group_t】

定义	类型	说明
start_chan	int32_t	开始通道号
stop_chans	int32_t*	终止通道号数组指针
chan_count	int32_t	终止通道号数组大小

【Start-stop 分析双终止通道分组结构体-cinst_dual_stops_group_t】

定义	类型	说明
start_chan	int32_t	开始通道号
x_stop_chan	int32_t	X-终止通道号
y_stop_chan	int32_t	Y-终止通道号

【NCoins 分析通道分组结构体-cinst_ncoins_group_t】

定义	类型	说明
channels	int32_t*	符合通道号数组指针
chan_count	int32_t	符合通道号数组大小
virtual_twin	int32_t	虚拟时间窗

【频率分析稳定性指标类型定义-cinst_stability_metric_type_t】

定义	值	说明
CINST_STABILITY_METRIC_ADEV	0	Allan Deviation (ADEV)
CINST_STABILITY_METRIC_MDEV	1	Modified Allan Deviation (MDEV)
CINST_STABILITY_METRIC_TDEV	2	Time Deviation (TDEV)
CINST_STABILITY_METRIC_HDEV	3	Hadamard Deviation (HDEV)

【API 定义】

API 函数	参数	返回值	功能描述	说明
ci_get_single_cps(ci_handle_t handle, int32_t channel)	handle：对象句柄 channel:通道号	int-实时计数率的值	获取通道实时计数率	
ci_get_coins_cps(ci_handle_t handle, int32_t channel)	handle：对象句柄 channel:通道号	int-实时计数率的值	获取通道符合计数率	必须先设置 DATA MODE 为 Global Coins (全局符合模式)
ci_output_single_cps(ci_handle_t handle, int32_t duration, const char* folder_path, const char* file_name)	handle：对象句柄 duration：持续时间 folder_path：文件夹路径 file_name：文件名	int-状态码	保存通道实时计数率到文件	<duration> 采集时间：单位-秒 <file_path> 未设置情况下，默认是 output 目录 <file_name> 未明确后缀名情况下，默认后缀名为“.csv”
ci_stop_singlecps_recording(ci_handle_t handle)	handle：对象句柄	int-状态码	停止保存实时计数率	

ci_output_coins_cps(ci_handle_t handle, int32_t duration, const char* folder_path, const char* file_name)	handle：对象句柄 duration：持续时间 folder_path：文件夹路径 file_name：文件名	int-状态码	保存通道符合计数率到文件	<duration> 采集时间：单位-秒 <file_path> 未设置情况下，默认是 output 目录 <file_name> 未明确后缀名情况下，默认后缀名为“.csv”
ci_stop_coinscps_recording(ci_handle_t handle)	handle：对象句柄	int-状态码	停止保存符合计数率	
ci_tof_analysis(ci_handle_t handle, int32_t duration, int32_t ch1, int32_t ch2, int32_t coins_time_window, int32_t avg_times, const char* folder_path, const char* file_name)	handle：对象句柄 duration：持续时间 ch1：通道 1 通道号 ch2：通道 2 通道号 coins_time_window：符合时间窗值 avg_times:开启平均触发模式后的平均次数 folder_path：文件夹路径 file_name：文件名	int-状态码	双通道 TOF 时间分析	<folder_path> 如果不设置，默认是在 output 目录 <file_name> 如果不设置，不保存文件 <avg_times> 如果设置为 0，默认是单次触发模式
ci_enable_cross_correlation_analysis(ci_handle_t handle, int32_t virtual_twin_ps, int32_t bin_width_ps, int32_t dcr1,	handle：对象句柄 virtual_twin_ps：虚拟时间窗（单位：ps） bin_width_ps：BIN 宽度（单位：ps） dcr1：通道 1 的暗计数率（CPS） dcr2：通道 2 的暗计数率	int-状态码	开启互关联分析	

int32_t dcr2)	（CPS）			
ci_fetch_tof_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 Tof Process 数据	返回扁平 int64 数组，格式为：<tdiff, counts>
ci_fetch_cross_correlation_data(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	double*	实时读取互关联分析数据	返回扁平 double 数组，格式为：<tdiff, g2>
ci_stop_tof_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 TOF 分析	
ci_startstop_analysis(ci_handle_t handle, int32_t duration,const cinst_chan_group_t* chan_groups, int32_t chan_groups_count, const cinst_dual_stops_group_t* dual_stops_groups, int32_t dual_stops_groups_count, int32_t coins_time_window, const char* folder_path, const char* file_name)	handle：对象句柄 duration：持续时间 chan_groups：通道分组指针 chan_groups_count：通道分组数量 dual_stops_groups：双终止通道分组指针 dual_stops_groups_count:双终止通道分组数量 coins_time_window：符合时间窗值 folder_path：文件夹路径 file_name：文件名	int-状态码	开始终止通道时间差分析	<chan_groups> cinst_chan_group_t 结构体数组指针 <dual_stops_groups> cinst_dual_stops_group_t结构体数组指针
ci_fetch_startstop_processdata(ci_handle_t handle, int32_t start_chan, int32_t stop_chan, int32_t* out_data_points_size)	handle：对象句柄 start_chan：开始通道号 stop_chan：终止通道号 out_data_points_size:返回数值对的大小（请注意：不	int64_t*	实时读取 Start-stop Process 数据	返回扁平 int64 数组，格式为：<tdiff, counts>

	是数组长度）			
ci_fetch_startstop_average_processdata(ci_handle_t handle, int32_t start_chan, int32_t* out_data_points_size)	handle：对象句柄 start_chan：开始通道号 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 Start-stop Average Process 数据	返回扁平 int64 数组，格式为：<tdiff, counts>
ci_fetch_startstop_dualstops_rawdata(ci_handle_t handle, int32_t start_chan, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 start_chan：开始通道号 out_data_points_size:返回数值对的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取 Start-stop dual stops RAW 数据	返回扁平 int64 数组，格式为：<x_stop_tdiff, y_stop_tdiff>
ci_stop_startstop_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止开始终止通道时间差分析	
ci_ncoins_analysis(ci_handle_t handle, int32_t duration, const cinst_ncoins_group_t* ncoins_groups, int32_t ncoins_groups_count,int32_t coins_time_window,const char* folder_path,const char* file_name)	handle：对象句柄 duration：持续时间 ncoins_groups：符合计数的通道分组指针 ncoins_groups_count：通道分组的大小 coins_time_window：符合时间窗值 folder_path：文件夹路径 file_name：文件名	int-状态码	N 重符合计数分析	<ncoins_groups> cinst_ncoins_group_t 结构体数组指针
ci_fetch_ncoins_rawdata(ci_handle_t handle, const	handle：对象句柄 ncoins_group：符合计数的通道组	int64_t*	实时读取 N 重符合计数分析的	返回扁平 int64 数组，格式为：

cinst_ncoins_group_t ncoins_group, int32_t* out_data_points_size, int32_t timeout_ms)	out_data_points_size:返回数值组的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）		RAW 过程数据	<coins_index, start_ch, start_timestamp, end_ch, end_timestamp>
ci_fetch_ncoins_cps(ci_handle_t handle, const cinst_ncoins_group_t ncoins_group, int32_t* out_data_points_size)	handle：对象句柄 ncoins_group：符合计数的通道组 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 N 重符合计数分析的 CPS 数据	返回扁平 int64 数组，格式为：<time, cps>
ci_stop_ncoins_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 N 重符合计数分析	
ci_autocorrelation_analysis(ci_handle_t handle, int32_t duration, int32_t ch, double min_tau, double max_tau, int bin_num, int edge_type, const char* folder_path, const char* file_name)	handle：对象句柄 duration：持续时间 ch：通道号 min_tau：最小 tau 值 max_tau：最大 tau 值 bin_num：BIN 数量 edget_type：边沿类型（上升沿还是下降沿） folder_path：文件夹路径 file_name：文件名	int-状态码	单通道自关联函数（G2）分析	
ci_fetch_autocorrelation_data(ci_handle_t handle,	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	double*	实时读取自关联分析的分析数据	返回扁平 double 数组，格式为：<tdiff, g2>

int32_t* out_data_points_size);				
ci_stop_autocorrelation_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止自关联函数分析	
ci_tot_analysis(ci_handle_t handle, int32_t duration, int32_t ch, const char* folder_path, const char* file_name)	handle：对象句柄 duration：持续时间 ch：通道 folder_path：文件夹路径 file_name：文件名	int-状态码	过阈时间分析	
ci_fetch_tot_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 TOT 分析数据	返回扁平 int64 数组，格式为:<tot, counts>
ci_stop_tot_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 TOT 分析	
ci_period_analysis(ci_handle_t handle, int32_t duration, int32_t decades,int32_t decade_points,int32_t average_num,int32_t analysis_length,double fnom_hz,int32_t sampling_interval_ms,int32_t sampling_num,const	handle：对象句柄 duration：持续时间 decades：数量级大小 decade_points:每个数量级的点数 average_num：平均计算次数 analysis_length：分析数据长度 fnom_hz：标称频率（Hz） sampling_interval_ms：采样间隔时间 sampling_num：采样数量 folder_path：文件夹路径 file_name：文件名	int-状态码	频率分析	周期数据采样，只是为了显示周期波形，或者进行定量分析 <sampling_interval_ms> 采样间隔，单位：ms <sampling_num> 如果设为 0，采用默认的采样周期数量 100

char* folder_path,const char* file_name)				
ci_fetch_period_rawdata(ci_handle_t handle, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取频率分析的周期数据	返回扁平 int64 数组，格式为：<index, period>
ci_fetch_period_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取频率分析的分析数据	返回扁平 int64 数组，格式为：<period, counts>
ci_fetch_period_stability_m etric(ci_handle_t handle, cinst_stability_metric_type _t metric_type, int32_t* out_data_points_size);	handle：对象句柄 metric_type：稳定性指标类型 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	double*	实时读取频率分析的稳定性指标数据	返回扁平 double 数组，格式为：<tau, value>
ci_stop_period_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止频率分析	
ci_ad_analysis(ci_handle_t handle, int32_t duration, int32_t ch, int32_t ad_integration_point, int32_t sampling_interval_ms, int32_t sampling_num, const char* folder_path, const char* file_name)	handle：对象句柄 duration：持续时间 ch：通道号 ad_integration_point: A/D 面积积分值 sampling_interval_ms：采样间隔时间 sampling_num：采样数量 folder_path：文件夹路径 file_name：文件名	int-状态码	A/D 信号分析	<ch> 必须是 A/D 通道 <sampling_interval_ms> 采样间隔，单位：ms <sampling_num> 如果小于等于 0，全采样

ci_fetch_ad_rawdata(ci_handle_t handle, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取 A/D RAW 数据	返回扁平 int64 数组，格式为：<sampling_points, adc>
ci_stop_ad_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 A/D 信号分析	
ci_es_analysis(ci_handle_t handle, int32_t duration,, int32_t ch, int32_t ad_integration_point, int32_t ad_area_scale, const char* folder_path, const char* file_name)	handle：对象句柄 duration：持续时间 ch：通道 ad_integration_point: A/D 面积积分值 ad_area_scale: A/D 面积积分缩放类型 folder_path：文件夹路径 file_name：文件名	int-状态码	能谱分析	<ch> 必须是 A/D 通道
ci_fetch_es_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 ES 分析数据	返回扁平 int64 数组，格式为：<adc, counts>
ci_stop_es_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止能谱分析	
ci_conform_es_analysis(ci_handle_t handle, int32_t duration, int32_t ch1, int32_t ch2, int32_t ad_integration_point, int32_t ad_area_scale, int32_t energy_window_min1, int32_t energy_window_max1, int32_t energy_window_min2, int32_t energy_window_max2, int32_t	handle：对象句柄 duration：持续时间 ch1：通道 1 ch2：通道 2 ad_integration_point: A/D 面积积分值 ad_area_scale: A/D 面积积分缩放类型 energy_window_min1：能量窗口最小值 1 energy_window_max1：能量窗口最大值 1 energy_window_min2：能量窗口最小值 2	int-状态码	符合能谱分析	<ch1> 必须是 A/D 通道 <ch2> 必须是 A/D 通道 <energy_window_min1> <energy_window_max1> <energy_wi

coins_time_window, int32_t avg_times,　const char* folder_path, const char* file_name)	energy_window_max2：能量窗口最大值 2 coins_time_window：符合时间窗值 avg_times:开启平均触发模式后的平均次数 folder_path：文件夹路径 file_name：文件名			ndow_min2 > <energy_window_max2> 能量窗的有效值范围：0-16384
ci_fetch_conform_es_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 CONFORMES 分析数据	返回扁平 int64 数组，格式为：<tdiff, counts>
ci_fetch_conform_es_area1_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 CONFORMES Area1 分析数据（低通道号）	返回扁平 int64 数组，格式为：<adc, counts>
ci_fetch_conform_es_area2_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 CONFORMES Area2 分析数据（高通道号）	返回扁平 int64 数组，格式为：<adc, counts>
ci_stop_conform_es_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止符合能谱分析	
ci_clear_analysis_data(ci_handle_t handle)	handle：对象句柄	int-状态码	清除数据分析缓存	

9.1.12 其他辅助接口

【API 定义】

API 函数	参数	返回值	功能描述	注意事项
ci_free_char_data(c	ptr：内存指针	-	释放 API 分配的	主要针对 ci_fetch_rawdata 返

har* ptr)			Char 内存指针	回的内存指针，需要手动进行释放
ci_free_int64_data(int64_t* ptr)	ptr：内存指针	-	释放 API 分配的 Int64 内存指针	主要针对数据分析相关 API 返回的 Int64 内存指针，需要手动进行释放
ci_free_double_data(double* ptr)	ptr：内存指针	-	释放 API 分配的 Double 内存指针	主要针对数据分析相关 API 返回的 Double 内存指针，需要手动进行释放
ci_copy_data(char* dest, uint64_t src_addr, int size)	dest：目的内存指针 src_addr：API 分配的内存指针值	-	拷贝 API 分配的内存到目的内存地址	主要给 LabVIEW 调用
ci_copy_int64_data(int64_t* dest, uint64_t src_addr, int size)	dest：目的 Int64 内存指针 src_addr：API 分配的内存指针值	-	拷贝 API 分配的内存到目的内存地址	主要给 LabVIEW 调用
ci_int64_pairs_to_string(char* dest, int32_t dest_size, const int64_t* data, int32_t data_points_size)	dest：目的内存指针 dest_size：目的内存大小 data：源数据地址 data_points_size：源数据组大小	int-实际字节数	转换 int64 数据到字符串	Int64 是数据分析 API 返回的一维数据，每 2 位数字表示一组 pair 数据。
ci_double_pairs_to_string(char* dest, int32_t dest_size, const double* data, int32_t data_points_size)	dest：目的内存指针 dest_size：目的内存大小 data：源数据地址 data_points_size：源数据组大小	int-实际字节数	转换 double 数据到字符串	Double 是数据分析 API 返回的一维数据，每 2 位数字表示一组 pair 数据。

9.1.13 通用注意事项

1) 状态码：

定义	值	说明
CINST_SUCCESS	0	操作成功
CINST_FAIL	-1	操作失败
CINST_TRUE	1	布尔值 True
CINST_FALSE	0	布尔值 False

2) 线程安全：

- 所有 API 调用都是线程安全的；
- 数据采集和数据分析操作是异步的；
- 每次数据采集和分析完，建议手动调用 stop API。

9.2 C++ API

9.2.1 概述

ChronosInst 是一个用于高精度测量设备（比如：TDC 时间数字转换器设备）操作的 C++封装类，提供设备连接、配置、数据采集和分析等功能。该类基于 Qt 框架开发，支持信号槽机制和异步操作。

这份文档涵盖了 ChronosInst C++ API 的主要功能和使用方法，可用于开发基于该接口的应用程序。

9.2.2 初始化与设备连接

9.2.2.1 初始化

需要指定 API 配置文件路径，可以是绝对路径，也可以是相对路径。

如果是相对路径，会先在环境变量（CSP_BASE_PATH）指定目录下查找；如果没有找到，会在配置的 baseDir 目录下查找；最后，会在执行程序当前目录下查找。

【API 定义】

API	参数	参数说明	备注
ChronosInst (QString configPath, QString baseDir)	configPath	配置文件路径	确保文件存在
	baseDir	基础目录	相对路径，都会相对于这个基础目录（如果未设置 CSP_BASE_PATH 环境变量）

9.2.2.2 设备发现与连接

目前主要支持三种接口类型：

定义	值	说明
INST_USB3_DEV	1	USB3.0 接口
INST_UDP_DEV	2	UDP 千兆网口
INST_UDP_W_DEV	3	UDP 万兆网口

【API 定义】

API	参数	参数说明	备注
QList<INST_DEVICE_INFO> getDevices()	-	-	返回设备接口列表
bool connect(int connType)	connType	设备接口类型	连接设备 返回成功或失败

9.2.2.3 错误处理

基于 QT 信号槽机制的错误处理方式，可以获取最近的错误代码和错误信息。

【错误码定义】

定义	值	说明
ERROR_CONFIG_NOT_FOUND	2000	配置文件不存在
ERROR_INST_DEVICE_NOT_FOUND	2001	设备未找到
ERROR_INST_DEVICE_NOT_CONNECTED	2002	设备未连接
ERROR_INST_DEVICE_ALREADY_RUNNING	2003	设置已运行

ERROR_INST_START_PROTOCOL_FRAMEWORK	2004	启动协议层失败
ERROR_INST_PROTOCOL_FRAMEWORK_NOT_RUNNING	2005	协议层未运行
ERROR_INST_PROTOCOL_FRAMEWORK_INTERNAL	2006	协议层内部错误
ERROR_INST_INVALID_ACQ_TYPE	2007	错误的数据采集方式
ERROR_INST_ACQ_DATA_FAILED	2008	数据采集失败
ERROR_INST_PROCESS_ANALYSIS_FAILED	2009	数据分析失败
ERROR_INST_DATA_PROCESS_NOT_STOPPED	2010	上一次数据分析任务未完成
ERROR_INST_INVALID_CALIBRATION_PARAM	2011	错误的标定参数
ERROR_INST_CALIBRATION_FAILED	2012	设备标定失败
ERROR_INST_CONVERT_BINFILE_FAILED	2013	转换 BIN 文件失败
ERROR_INST_INVALID_PARAMETER	2014	错误的参数
ERROR_INST_TERMINATION_CONFIG_UNSUPPORTED	2015	不支持阻抗参数配置
ERROR_INST_ENABLE_CHANNEL_FAILED	2016	通道使能失败
ERROR_INST_DISABLE_CHANNEL_FAILED	2017	通道禁能失败
ERROR_INST_SET_CHANNELS_ENABLED_FAILED	2018	设置所有通道是否使能失败

【API 定义】

API	参数	参数说明	备注
void pf_errorOccurred(INST_ERROR error, const QString& description)	INST_ERROR	错误码	请详见前面错误码定义
	description	错误信息	

9.2.3 设备自检

设备使用前，可以先进行设备自检操作，设置各通道和全局配置的初始默认值。设备自检后，在 selfcheck 目录下会生成自检日志文档，可以查看自检状态。

【API 定义】

API	参数	参数说明	备注
bool selfCheck()	-	-	设备自检接口

			返回成功或失败

9.2.4 通道配置

9.2.4.1 DAC 设置

可以单独设置各个通道的 DAC 值（有效范围：-1500 到 1500，超出会失败）。

注意事项：

- 参考通道不支持设置 DAC 值；
- 如果 status 值为 1，说明配置值和实际生效值不相等，value 返回实际生效值 （后续配置也是同样逻辑）。

【API 定义】

API	参数	参数说明	备注
INST_RESULT<double> setDAC(int channel, double value)	channel	通道号	设置和获取 DAC 值 返回 INST_RESULT<double> status - 状态 value - DAC 值
	value	DAC 值	
INST_RESULT<double> getDAC(int channel)	channel	通道号	

9.2.4.2 阻抗设置

可以单独设置各个通道的阻抗值（有效值：50 Ohm 和 1M Ohm）。

注意事项：

- 仅部分型号支持设置通道 Termination 阻抗值（Venus Ultra 系列）；
- 参考通道不支持设置 Termination 阻抗值。

阻抗类型（INST_TERM_TYPE）：

定义	值	说明
INST_TERM_50	0	阻抗 50 Ohm（默认值）
INST_TERM_1M	1	阻抗 1M Ohm

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setTermination(int channel, int value)	channel	通道号	设置和获取 Termination 阻抗类型值 返回 INST_RESULT<int> status - 状态 value - 阻抗类型值
	value	阻抗类型值	
INST_RESULT<int> getTermination(int channel)	channel	通道号	

9.2.4.3 迟滞电压设置

设置通道的迟滞电压类型值（有效值范围：0-3，超出会失败）。

注意事项：

- 参考通道不支持设置迟滞电压。

通道迟滞电压类型（INST_HTS_TYPE）：

定义	值	说明
INST_HTS_30_MV	0	迟滞电压 30mV（默认值）
INST_HTS_40_MV	1	迟滞电压 40mV
INST_HTS_70_MV	2	迟滞电压 70mV
INST_HTS_1_MV	3	迟滞电压 1mV

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setHysteresis(int channel, int value)	channel	通道号	设置和获取通道迟滞电压类型值 （详见 INST_HTS_TYPE 类型定义） 返回 INST_RESULT<int> status - 状态 value - 迟滞电压类型值
	value	迟滞电压类型值	
INST_RESULT<int> getHysteresis(int channel)	channel	通道号	

9.2.4.4 通道间延迟设置

设置各个通道的时间延迟值（有效值范围：0-1000000ps， 超出会失败）。

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setInterDelay(int channel, int value)	channel	通道号	设置和获取通道延迟时间 返回 INST_RESULT<int> status - 状态 value - 时间延迟值
	value	通道时间延迟值	
INST_RESULT<int> getInterDelay(int channel)	channel	通道号	

9.2.4.5 死时间设置

设置各个通道的死时间值（有效值范围：2-130000ps，超出会失败）。

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setDeadTime(int channel, int value)	channel	通道号	设置和获取通道死时间 返回 INST_RESULT<int> status - 状态 value - 死时间值
	value	通道死时间值	
INST_RESULT<int> getDeadTime(int channel)	channel	通道号	

9.2.4.6 A/D 面积积分设置

设置 A/D 通道（目前设备的奇数通道，是 A/D 通道）的面积积分值（有效值范围：0-65535，超出会失败）。

注意事项：

- 参考通道不支持设置 A/D 面积积分值。

【API 定义】

API	参数	参数说明	备注

INST_RESULT<int> setADIntegrationPoint(int channel, int value)	channel	通道号	设置和获取通道 A/D 面积积分值 返回 INST_RESULT<int> status - 状态 value - A/D 面积积分值
	value	通道 A/D 面积积分值	
INST_RESULT<int> getADIntegrationPoint(int channel)	channel	通道号	

9.2.4.7 边沿触发类型设置

设置通道的边沿触发类型：

定义	值	说明
INST_RISING_EDGE	0	上升沿
INST_FALLING_EDGE	1	下降沿
INST_PMIDDLE	2	正中间
INST_NMIDDLE	3	负中间

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setEdgeType(int channel, int value)	channel	通道号	设置和获取通道边沿触发类型值 返回 INST_RESULT<int> status - 状态 value - 边沿触发类型值
	value	通道边沿触发类型值	
INST_RESULT<int> getEdgeType(int channel)	channel	通道号	

9.2.4.8 通道使能设置

可以对通道设置使能/禁能。禁能后，该通道关闭，不再输出测量数据。

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> enableChannel(int channel)	channel	通道号	设置通道使能/禁能 返回 INST_RESULT<int> status - 状态
INST_RESULT<int> disableChannel(int channel)	channel	通道号	

			value - 使能状态值
INST_RESULT<bool> isChannelEnabled(int channel)	channel	通道号	获取通道使能状态 INST_RESULT<bool> status - 状态 value - 是否使能
bool enableAllChannels()	-	-	设置所有通道使能，返回成功或失败
bool disableAllChannels()	-	-	设置所有通道禁能，返回成功或失败

9.2.5 全局配置

9.2.5.1 板卡 ID 设置

设置当前设备板块的 ID 号，主要用于设备并联时区分设备使用（有效值范围：0-255，超出会失败）。

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setBoardId(int newBoardId)	newBoardId	板卡新的 ID 号	设置和获取板号 返回 INST_RESULT<int> status - 设置状态 value - 板卡 ID 号
Int getBoardId()	-	-	返回板卡 ID 号

9.2.5.2 TDC 时钟来源设置

设置 TDC 的时钟来源，切换后，设备需要重新校准，耗时较长（一般需要 20 秒

左右）：

定义	值	说明
INST_INTERNAL_CLOCK	0	本地时钟
INST_EXTERNAL_CLOCK	1	外部时钟

【API 定义】

<table>
<tr><th>API</th><th>参数</th><th>参数说明</th><th>备注</th></tr>
<tr><td>INST_RESULT<int>
setTdcClockSource(int srcType)</td><td>srcType</td><td>时钟来源类型</td><td rowspan="2">设置和获取 TDC 时钟来源
返回 INST_RESULT<int>
status - 状态
value - 时钟来源类型值</td></tr>
<tr><td>INST_RESULT<int>
getTdcClockSource()</td><td>-</td><td>-</td></tr>
</table>

9.2.5.3 数据模式设置

设置当前的数据模式：

定义	值	说明
INST_DATA_MODE_TIMESTAMP	0	时间模式
INST_DATA_MODE_TIME ZONE	1	时区模式
INST_DATA_MODE_AD	2	A/D 模式
INST_DATA_MODE_AREA	3	面积测量模式
INST_DATA_MODE_REF_COINS	4	参考符合模式
INST_DATA_MODE_GLOBAL_COINS	5	全局符合模式
INST_DATA_MODE_TOT	6	过阈时间测量模式
INST_DATA_MODE_AREA_COINS	7	能谱-全局符合模式
INST_DATA_MODE_DUAL_TIMESTAMP	8	双沿时间模式
INST_DATA_MODE_PERIOD	9	周期测量模式
INST_DATA_MODE_SINGLE_COINS	11	单符合模式
INST_DATA_MODE_GLOBAL_COINS_TIMESTAMP	12	全局符合（带时间戳）模式

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setDataMode(int dataMode)	dataMode	数据模式类型	设置和获取数据模式 返回 INST_RESULT<int> status - 状态 value - 数据模式类型值
INST_RESULT<int> getDataMode()	-	-	

9.2.5.4 符合时间窗设置

设置符合时间窗数值（有效值范围：0-16000000ps，超出会失败）。

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setCoinsTimeWindow(int coinsTimeWindow)	coinsTimeWindow	符合时间窗值	设置和获取符合时间窗值 返回 INST_RESULT<int> status - 状态 value - 符合时间窗值
INST_RESULT<int> getCoinsTimeWindow()	-	-	

9.2.5.5 测试模式设置

设置当前信号源模式，主要供内部仿真和测试使用，正常使用采用 RAW 模式：

定义	值	说明
INST_TEST_MODE_RAW	0	RAW 原始数据模式
INST_TEST_MODE_ALL_0	1	全 0 模式

INST_TEST_MODE_ALL_1	2	全 1 模式
INST_TEST_MODE_TOGGLE	3	转换模式
INST_TEST_MODE_INCREASE	4	递增模式
INST_TEST_MODE_DECREASE	5	递减模式
INST_TEST_MODE_1M_TRIGGER	6	1MHz 内部触发模式

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setTestMode(int testMode)	testMode	测试模式类型值	设置和获取测试模式类型 返回 INST_RESULT<int> status - 状态 value - 测试模式类型值
INST_RESULT<int> getTestMode()	-	-	

9.2.5.6 数据带宽设置

设置数据带宽:

定义	值	说明
INST_TEST_DATA_BANDWIDTH_100M	0	100M 数据带宽
INST_TEST_DATA_BANDWIDTH_1000M	1	1000M 数据带宽
INST_TEST_DATA_BANDWIDTH_3000M	2	3000M 数据带宽
INST_TEST_DATA_BANDWIDTH_6400M	3	6400M 数据带宽

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setTestDataBandwidth(int dataBandwidth)	dataBandwidth	数据带宽类型值	设置和获取测试数据带宽类型 返回 INST_RESULT<int> status - 状态 value - 数据带宽类型值
INST_RESULT<int> getTestDataBandWidth()	-	-	

9.2.5.7 A/D 面积积分缩放类型设置

设置 A/D 面积积分缩放类型:

定义	值	说明
INST_AREA_SCALE_1	0	1 倍
INST_AREA_SCALE_1_2	1	1/2 倍
INST_AREA_SCALE_1_4	2	1/4 倍
INST_AREA_SCALE_1_8	3	1/8 倍
INST_AREA_SCALE_1_16	4	1/16 倍
INST_AREA_SCALE_1_32	5	1/32 倍
INST_AREA_SCALE_1_64	6	1/64 倍
INST_AREA_SCALE_1_128	7	1/128 倍

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setADAreaScale(int adAreaScale)	adAreaScale	面积积分缩放类型值	设置和获取积分缩放类型 返回 INST_RESULT<int> status - 状态 value - 面积积分缩放类型值
INST_RESULT<int> getADAreaScale()	-	-	

9.2.5.8 数据采集方式设置

设置数据采集方式:

定义	值	说明
INST_ACQ_SOFTWARE	0	软件方式
INST_ACQ_HARDWARE	1	硬件方式

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setAcqType(int acqType)	acqType	数据采集方式	设置和获取数据采集方式值 返回 INST_RESULT<int> status - 状态 value - 数据采集方式值

9.2.5.9 设备信息获取

可以获取以下设备信息：

API	说明	备注
int getChannelNum()	获取设备有效通道数	不包括设备的参考通道
Int getDeviceType()	获取设备类型	0x1: Venus（TDC） 0x2: Mercury（多道分析仪）
int getDeviceMode()	获取设备型号	0x1: Venus Lite-32/Mercury-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0x10 - Venus Lite 4 0x11 - Venus Lite 2
INST_RESULT<QString> getProductSN()	获取设备产品序列号	INST_RESULT<QString> status - 状态 value - 面积积分缩放类型值

INST_DEVICE_STATUS getDeviceStatus()	获取设备的状态	获取设备核心温度和各电路电流值(mA)
		INST_DEVICE_STATUS{
		int temperature; // 核心温度
		float current_1; // 电路电流 mA
		float current_2;
		float current_3;
		float current_4;
		float current_5;
		float current_6;
		float current_7;
		float current_8;
		}

int getFanSpeed()	获取风扇转速（单位：RPM）	

const QString getFWVersion()	获取固件版本号	
const QString getSWVersion()	获取软件版本号	

9.2.5.10 复位控制

主要包括 TDC 复位和系统全局复位。

注意事项：

- 系统复位会对设备进行整体的复位，耗时较长（一般在 10s 左右），请谨慎操作。

API	参数	参数说明	备注
bool tdcResetn()	-	-	TDC 复位 返回成功或失败
bool systemResetn()	-	-	系统复位 返回成功或失败

9.2.6 网络配置

9.2.6.1 千兆网 IP 地址设置

设置千兆网口的 IP 地址。

注意事项：

- 本机 IP 和设备 IP 需要在同一个网段；
- 修改 IP 地址后，请重连设备进行操作。

【API 定义】

API	参数	参数说明	备注
INST_NET_RESULT<QString> setIPs(const QString& local_ip, const QString& device_ip)	local_ip	本地 IP 地址	设置和获取千兆网 IP 地址 返回
	device_ip	设备 IP 地址	

INST_NET_RESULT<QString> getIPs()	-	-	INST_NET_RESULT<QString> status - 状态 local_value - 本地 IP 地址 device_value - 设备 IP 地址

9.2.6.2 千兆网端口设置

设置千兆网口的网络端口。

注意事项：

- 注意本地端口的占用情况，请不要设置已被占用的端口号；
- 修改网络端口后，请重连设备进行操作。

【API 定义】

API	参数	参数说明	备注
INST_NET_RESULT<int> setNetPorts(int local_port, int device_port)	local_port device_port	本地网络端口 设备网络端口	设置和获取千兆网端口 返回 INST_NET_RESULT<QString> status - 状态 local_value - 本地网络端口 device_value - 设备网络端口
INST_NET_RESULT<int> getNetPorts()	-	-	

9.2.6.3 万兆网 IP 设置

设置万兆网口的 IP 地址。

注意事项：

- 本机 IP 和设备 IP 需要在同一个网段；
- 修改 IP 地址后，请重连设备进行操作。

【API 定义】

API	参数	参数说明	备注
INST_NET_RESULT<QString>	local_ip	本地 IP 地址	设置和获取万兆网 IP

setWIPs(const QString& local_ip, const QString& device_ip)	device_ip	设备 IP 地址	返回 INST_NET_RESULT<QString> status - 状态 local_value - 本地 IP 地址 device_value - 设备 IP 地址
INST_NET_RESULT<QString> getWIPs()	-	-	

9.2.6.4 万兆网端口设置

设置万兆网口的网络端口。

注意事项：

- 注意本地端口的占用情况，请不要设置已被占用的端口号；
- 修改网络端口后，请重连设备进行操作。

【API 定义】

API	参数	参数说明	返回
INST_NET_RESULT<int> setWNetPorts(int local_port, int device_port)	local_port	本地网络端口	设置和获取万兆网端口 返回 INST_NET_RESULT<QString> status - 状态 local_value - 本地网络端口 device_value - 设备网络端口
	device_port	设备网络端口	
INST_NET_RESULT<int> getWNetPorts()	-	-	

9.2.6.5 千兆网 MAC 地址设置

设置设备千兆网口 MAC 地址。

注意事项：

- 在一个网段内，确保设置的 MAC 地址是唯一的；
- 修改 MAC 地址后，一般需要重启交换机或路由器，清空 MAC 地址缓存，重新连接设备。

【API 定义】

API	参数	参数说明	返回
INST_MAC_RESULT setMAC(const QString& mac)	mac	网口 MAC 地址	设置和获取千兆网口 MAC 地址 返回 INST_NET_RESULT-MAC status - 状态 device-mac - 设备网口 MAC 地址
INST_MAC_RESULT getMAC()	-	-	

9.2.6.6 万兆网 MAC 地址设置

设置设备万兆网口的 MAC 地址。

注意事项:

- 在一个网段内，确保设置的 MAC 地址是唯一的;
- 修改 MAC 地址后，一般需要重启交换机或路由器，清空 MAC 地址缓存，重新连接设备。

【API 定义】

API	参数	参数说明	备注
INST_MAC_RESULT setWMAC(const QString& mac)	mac	网口 MAC 地址	设置和获取万兆网口 MAC 地址 返回 INST_NET_RESULT-MAC status - 状态 device-mac - 设备网口 MAC 地址
INST_MAC_RESULT getWMAC()	-	-	

9.2.7 数据采集

指定数据模式下的 RAW 数据采集接口。

注意事项:

- 数据采集 API 是异步的，API 调用后，不会阻塞线程，会立即返回，需要程序自己处理等待逻辑;
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日

志中进行查看；

- 提供了实时获取当前数据的接口，可以读取数据到内存进行实时处理；
- 可以调用 stopAcq API 去强制停止当前的数据采集任务；如果不调用，采集时间到了后，系统会自动调用 stopAcq API，停止数据采集；
- 启用十六进制格式输出，写文件速率较慢（默认二进制数据格式，文件写入更高效），耗时较长，请注意数据文件大小。

【API 定义】

API	参数	参数说明	备注
bool enableHexDataFormat()	-	-	启用十六进制格式 返回成功或失败
bool disableHexDataFormat()	-	-	禁用十六进制格式（默认二进制格式） 返回成功或失败
bool isHexDataFormat()	-	-	当前是否启用十六进制格式 返回是或否
bool setRawDataCache(int bufferSizeMB)	bufferSizeMB	缓存大小（单位：MB）	设置 RAW 数据缓存（支持实时从内存读取数据） 返回成功或失败

bool acqRawData(int dataMode, int duration，const QString& folderPath, const QString& fileName, int maxVolumeSizeMB = -1, bool forceClose = false)	dataMode	数据模式（详见前面的数据模式定义）	开始数据采集（异步模式，立即返回） 返回成功或失败。 如果设置了输出文件名，会启用数据文件保存。未明确后缀名情况下，如果启用十六进制格式输出，后缀名为“.dat”;如果未启用，默认是二进制格式输出，后缀名为“.bin”。
	duration	数据采集时间（单位：秒）	
	folderPath	输出文件目录设置（默认是 raw 目	

		录）	
	fileName	输出文件名称（如果不指定，不保存文件）	
	maxVolumeSizeMB	文件分卷大小（默认不分卷，单位：MB）	
	forceClose	是否强制停止文件保存（默认是false，等待内存中数据全部保存）	
bool fetchRawData(std::vector<char>& outData, int timeout_ms)	outData	实时返回的 Raw 数据	实时读取 Raw 数据 返回成功或失败
	timeout_ms	超时时间（单位：ms）	
bool stopAcq()	-	-	停止数据采集 返回成功或失败
bool convertToHex(const QString& binFile, const QString& hexFile, bool isAsync = false)	binFile	二进制原始数据文件路径	转换二进制格式文件到十六进制格式 （请确认目标文件是二进制文件格式）
	hexFile	输出的十六进制文件路径	如果输入为空，默认生成的十六进制文件在二进制文件同级目录下
	isAsync	是否异步处理	如果异步处理，方法立即返回；需要调用方侦听转换状态信号
void stopAsyncConvert()	-	-	如果采用异步处理模式，可以调用停止方法，停止转换操作

9.2.8 数据分析

9.2.8.1 强度分析

各通道实时计数率和符合计数率的 API 接口，数据可以文件格式输出，也可以实时获取。

注意事项：

- 文件输出 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的任务；如果不调用，时间到了后，系统会自动调用 Stop API，停止任务；
- 输出符合计数率前，请先设置 DATA MODE 为 GLOBAL_COINS（全局符合模式）。

【API 定义】

API	参数	参数说明	备注
bool outputSingleCPS(int duration,const QString& folderPath, const QString& fileName)	duration	数据采集时间(单位：秒)	输出各通道实时计数率到文件 返回成功或失败
	folderPath	数据文件输出目录（不设置，默认是 output 目录）	
	fileName	数据文件名称（未明确后缀名情况下，默认后缀名为“.csv”）	
bool stopSingleCPSRecording()	-	-	停止实时计数率保存任务 返回成功或失败
bool outputCoinsCPS(int duration,const QString& folderPath, const QString& fileName)	duration	数据采集时间(单位：秒)	输出各通道符合计数率到文件 返回成功或失败
	folderPath	数据文件输出目录（不设置，默认是 output 目录）	

	fileName	数据文件名称（未明确后缀名情况下，默认后缀名为“.csv”）	
bool stopCoinsCPSRecording()	-	-	停止符合计数率保存任务 返回成功或失败
int getSingleCPS(int channel)	channel	通道号	获取指定通道的实时计数率
int getCoinsCPS(int channel)	channel	通道号	获取指定通道的符合计数率

9.2.8.2 TOF 分析（双通道时间差分析）

对指定的两个通道进行 TOF 时间分析，数据以文件格式输出，也可以实时从内存读取分析数据。

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool tofAnalysis(int duration, const INST_TOF_OPTIONS& options, const QString& folderPath = QString(), const QString& fileName = QString())	duration	数据采集时间	单位：秒
	options	INST_TOF_OPTIONS{ ch1, ch2, coins_time_window, enable_avg_mode	<ch1> 通道 1 通道号 <ch2> 通道 2 通道号 <coins_time_window> 符合时间窗值

		, avg_times }	<enable_avg_mode> 是否启用平均触发模式 <avg_times> 启用平均触发模式后，平均次数设置
	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool enableCrossCorrelationAnalysis(const INST_CROSS_CORR_CFG& cfg)	cfg	启用互关联分析	返回成功或失败
bool fetchTofProcessData(QVector<QPointF>& outData)	outData	返回 TOF 分析数据	QPointF 格式 X-tdiff 值 Y-counts 数量
bool fetchCrossCorrelationData(QVector<QPointF>& outData)	outData	返回互关联分析数据	QPointF 格式 X-tdiff 值 Y-g2 值
bool stopTofAnalysis()	-	-	返回成功或失败

9.2.8.3 开始-终止通道时间差分析

对指定的多个通道分组（一个起始通道和多个终止通道）进行时间差分析，数据以文件格式输出。

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日

志中进行查看；

- 对于双终止通道分析，必须通道分组中，终止通道多于 2 个，并且指定不同的 X-终止通道和 Y-终止通道；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool startStopAnalysis(int duration, const INST_START_STOP_OPTIONS& options, const QString& folderPath, const QString& fileName)	duration	数据采集时间	单位：秒
	options	INST_DECAY_TIME_OPTIONS{ start_ch, stop_ch, coins_time_window}	<start_ch> 起始通道号 <stop_ch> 终止通道号 <coins_time_window> 符合时间窗值
	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool fetchStartStopProcessData(int start_chan, int stop_chan, QVector<QPointF>& outData)	start_chan	开始通道号	
	stop_chan	终止通道号	
	outData	返回 Start-stop Process 数据	QPointF 格式 X-tdiff 值 Y-counts 数量
bool fetchStartStopAverageProcessData(int start_chan, QVector<QPointF>& outData)	start_chan	开始通道号	
	outData	返回 Start-stop Average Process 数据	QPointF 格式 X-tdiff 值 Y-counts 数量
bool fetchStartStopDualStopsRawData(i	start_chan	开始通道号	
	outData	返回 Start-stop Dual Stops	QPointF 格式

nt start_chan, QVector<QPointF>& outData, int timeout_ms)		Raw 数据	X-stop tdiff 值 Y-stop tdiff 值
	timeout_ms	超时时间（单位：ms）	
bool stopStartStopAnalysis()	-	-	返回成功或失败

9.2.8.4 N 重符合计数分析

对指定分组的多个通道进行符合计数分析，数据以文件格式输出，也可以实时从内存读取分析数据。

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool nCoinsAnalysis(int duration, const INST_NCOINS_OPTIONS& options, const QString& folderPath = QString(), const QString& fileName = QString())	duration	数据采集时间	单位：秒

	options	INST_NCOINS_OPTIONS{
		n_coins_groups;
		coins_time_window;
		}
		
		INST_NCOINS_GROUP{
		channels;
		virtual_twin;
		}
			N 重符合计数分析配置
			<n_coins_groups>
			符合分析的通道分组集合
			<coins_time_window>
			符合时间窗值
			
			N 重符合计数分析通道分组
			<channels>
			通道号组
			<virtual_twin>

			虚拟时间窗
	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
QString getNCoinsGroupName(const QSet<int>& channels)	channels	通道号组	返回内部规则生成的分组名
bool fetchNCoinsRawData(const QString& groupName, QVector<NCOINS_CHAN_PAIR>& outData, int timeout_ms)	groupName	分组名	

	outData	返回指定分组的符合计数数据	NCOINS_CHAN_PAIR{
			coins_index;
			(当前符合的 Index)
			start_ch;
			(开始通道号)
			start_timestamp;
			(开始通道的时间戳)
			end_ch;
			（终止通道号）
			end_timestampe;
			（终止通道的时间戳）
			}

	timeout_ms	超时时间（单位：ms）	
bool fetchNCoinsCPS(const QString& groupName, QVector<QPointF>& outData)	groupName	分组名	
	outData	返回指定分组的计数率 CPS 数据	QPointF 格式 X-time（s） Y-cps 值
bool stopNCoinsAnalysis()	-	-	返回成功或失败

9.2.8.5 单通道自关联函数分析

对指定通道进行自关联函数（G2）分析，数据以文件格式输出，也可以实时从

内存读取分析数据。

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool autoCorrelationAnalysis(int duration, const INST_AUTO_CORRELATION_OPTIONS& options, const QString& folderPath = QString(), const QString& fileName = QString())	duration	数据采集时间	单位：秒

	options	INST_AUTO_CORRELATION_OPTIONS{
		ch;
		min_tau;
		max_tau;
		bin_num;
		edge_type;
		}
			自关联函数分析配置
			<ch>
			分析的通道号
			<min_tau>
			最小 tau 值（单位：s）
			<max_tau>
			最大 tau 值（单位：s）
			<bin_num>
			BIN 数量
			<edge_type>
			分析边沿类型（上升沿或下降沿）

	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”

bool fetchAutoCorrelationData(QVector<QPointF>& outData)	outData	返回自关联函数分析数据	QPointF 格式 X-tau（s） Y-g2 值
bool stopAutoCorrelationAnalysis()	-	-	返回成功或失败

9.2.8.6 TOT 分析（过阈时间分析）

对指定通道的 TOT（过阈时间分析）分析，数据以文件格式输出。

注意事项:

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑;
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看;
- 可以调用 Stop API 去强制停止当前的分析任务;如果不调用,分析时间到后,系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool totAnalysis(int duration, int ch, const QString& folderPath, const QString& fileName)	duration	数据采集时间	单位：秒
	ch	通道号	
	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool fetchTotProcessData(QVector<QPointF>& outData)	outData	返回 TOT 分析数据	QPointF 格式 X-tot 值 Y-counts 数量
bool stopTotAnalysis()	-	-	返回成功或失败

9.2.8.7 频率分析

对通道接入的信号进行频率分析，数据以文件格式输出。

注意事项：

- 请确保只接入 1 路外部信号（只有前 2 个通道可以进行频率分析）；
- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool periodAnalysis(int duration, const INST_PERIOD_OPTIONS& options, const QString& folderPath, const QString& fileName)	duration	数据采集时间	单位：秒

	options	INST_PERIOD_OPTIONS{
		decades,
		decade_points,
		average_num,
		analysis_length,
		fnom_hz,
		sampling_interval,
		sampling_num
		}
			<decades>
			数量级大小
			<decade_points>
			每个数量级的点数
			<average_num>
			平均计算次数
			<analysis_length>
			分析数据长度
			<fnom_hz>
			标称频率（单位：Hz）
			<sampling_interval>
			抽样间隔时间
			<sampling_num>
			抽样数量，如果设为 0，默认 100。
			抽样数据只作为周期数据展示，或者定量

			分析。
	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool fetchPeriodRawData(QVector<QPointF>& outData, int timeout_ms)	outData	返回抽样周期数据	QPointF 格式 X-index Y-period 周期值(单位：ps)
	timeout_ms	超时时间（单位：ms）	
bool fetchPeriodProcessData(QVector<QPointF>& outData)	outData	返回频率分析周期统计数据	QPointF 格式 X-period 值 Y-counts 数量

INST_STABILITY_METRICS
 	
fetchAllPeriodStabilityMetrics()
	-	返回所有稳定性分析指标数据	INST_STABILITY_METRICS {
			adevs;
			tdevs;
			mdevs;
			hdevs;
			}

bool fetchPeriodStabilityMetric(INST_STABILITY_METRIC_TYPE metricType , QVector<QPointF>& outData)	metricType	稳定性分析指标类型	
	outData	返回指定类型的稳定性分析数据	QPointF 格式 X-tau(s) Y-value
bool stopPeriodAnalysis()	-	-	返回成功或失败

9.2.8.8 A/D 分析

对指定的通道进行 A/D 信号数据分析，数据以文件格式输出。

注意事项：

- 该接口仅对 A/D 通道有效（设备奇数通道），请确保外接信号；
- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool adAnalysis(int duration, const INST_AD_OPTIONS& options, const QString& folderPath, const QString& fileName)	duration	数据采集时间	单位：秒

	options	INST_AD_OPTIONS{
		ch,
		ad_integration_point,
		sampling_interval,
		sampling_num
		}
			<ch>
			A/D 通道的通道号
			<ad_integration_point>
			通道的面积积分值
			<sampling_interval>
			采样间隔，单位：ms
			<sampling_num>
			采样数量，如果设为 0，全采样

	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool fetchAdRawData(QVector<QPointF>& outData, int timeout_ms)	outData	返回 A/D Raw 数据	QPointF 格式 X-sampling_points 抽样点 Y-adc 信号 ADC 值
	timeout_ms	超时时间（单位：ms）	
bool stopAdAnalysis()	-	-	返回成功或失败

9.2.8.9 能谱分析

对指定通道进行能谱信号数据分析，数据以文件格式输出。

注意事项：

- 该接口仅对 A/D 通道有效（设备奇数通道），请确保外接信号；
- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool esAnalysis(int duration, const INST_ES_OPTIONS& options, const QString& folderPath, const QString& fileName,)	duration	数据采集时间	单位：秒

	options	INST_ES_OPTIONS{
		ch,
		ad_integration_point,
		ad_area_scale
		}
			<ch>
			A/D 通道的通道号
			<ad_integration_point>
			通道的面积积分值
			<ad_area_scale>
			A/D 面积积分缩放类型，请查看前面的详细定义

	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool fetchEsProcessData(QVector<QPointF> & outData)	outData	返回 ES Process 数据	QPointF 格式 X-adc 值 Y-counts 数量
bool stopEsAnalysis()	-	-	返回成功或失败

9.2.8.10 符合能谱分析

对指定的两个 A/D 通道进行时间符合能谱数据分析，数据以文件格式输出。

注意事项：

- 该接口仅对 A/D 通道有效（设备奇数通道），请确保外接信号；
- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 能量窗的有效值范围：0-16384；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool conformEsAnalysis(int duration, const INST_CONFORM_ES_OPTIONS& options, const QString& folderPath, const QString& fileName)	duration	数据采集时间	单位：秒

	options	INST_CONFORM_ES_OPTIONS{
		ch1,
		ch2,
		ad_integration_point,
		ad_area_scale,
		energy_window_min1,
		energy_window_max1,
		energy_window_min2,
		energy_window_max2,
		coins_time_window,
		enable_avg_mode,
		avg_times
		}
			<ch1>
			A/D 通道 1 的通道号
			<ch2>
			A/D 通道 2 的通道号
			<ad_integration_point>
			通道的面积积分值
			<ad_area_scale>
			A/D 面积积分缩放类型，请查看前面的详细定义
			<energy_window_min1>
			通道 1 最小能量窗设置
			<energy_window_max1>

			通道 1 最大能量窗设置 <energy_window_min2> 通道 2 最小能量窗设置 <energy_window_max2> 通道 2 最大能量窗设置 <coins_time_window> 符合时间窗值 <enable_avg_mode> 是否启用平均触发模式 <avg_times> 启用平均触发模式后，平均次数设置
	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool fetchConformEsProcessData(QVector<QPointF>& outData)	outData	返回 Conform ES Process 数据	QPointF 格式 X-tdiff 值 Y-counts 数量
bool fetchConformArea1ProcessData(QVector<QPointF>& outData)	outData	返回 Conform ES Area1 Process 数据（对应低通道号）	QPointF 格式 X-adc 值 Y-counts 数量
bool fetchConformArea2ProcessData(QVector<QPointF>& outData)	outData	返回 Conform ES Area2 Process 数据（对应高通道号）	QPointF 格式 X-adc 值 Y-counts 数量

bool stopConformEsAnalysis()	-	-	返回成功或失败

9.2.9 注意事项

1) **设备连接**：在执行任何操作前，必须先成功连接设备。
2) **错误处理**：建议始终连接错误信号以处理可能发生的异常情况。
3) **参数范围**：各配置参数有有效范围限制，超出范围的操作将失败。
4) **异步操作**：部分数据采集和分析操作是异步的，需要适当等待或使用回调机制。
5) **资源清理**：使用完成后，应正确断开设备连接。
6) **线程安全**：API 调用通常需要在主线程或特定线程中进行，请注意线程上下文。

9.2.10 示例代码

典型的应用代码示例如下：

```
int main(int argc, char *argv[])
{
    QCoreApplication app(argc, argv);

    // 1. 初始化 TDC
    ChronosInst tdc("tdcconfig.ini", "D:\\cinstapi");

    // 2. 连接错误信号
    QObject::connect(&tdc, &ChronosInst::pf_errorOccurred, ...);

    // 3. 获取并连接设备
    QList devices = tdc.getDevices();
    if(devices.size() > 0) {
        tdc.connect(devices.at(0).type);
    }

    // 4. 启动工作线程，执行配置和操作
    ...
     // 设置通道 DAC 值
```

```
    INST_RESULT<double> ret = tdc.setDAC(0, 100);    //  通道 0，值 100
    ret = tdc.setDAC(1, 100);    //  通道 1，值 100

    //  设置数据模式
    INST_RESULT<int> result = tdc.setDataMode(INST_DATA_MODE_GLOBAL_COINS);

    //  启用十六进制文件格式
    bool flag = tdc.enableHexFileFormat();

    //  采集原始数据
    flag = tdc.setRawDataCache(200);
    flag = tdc.acqRawData(INST_DATA_MODE_GLOBAL_COINS, 10, "raw", "test");

    //  实时读取数据
    std::vector<char> outData;
    flag = tdc.fetchRawData(outData, 500);

    //  等待数据采集完成
    QThread::sleep(10);

    //  停止采集
    flag = tdc.stopAcq();

    ...
    // 5. 进入事件循环
    return app.exec();
}
```

9.3 PYTHON API

9.3.1 概述

ChronosInst 是一个用于高精度测量设备（比如：TDC 时间数字转换器设备）操作的 Python 封装类，提供设备连接、配置、数据采集和分析等功能。

这份文档涵盖了 ChronosInst Python API 的主要功能和使用方法，可用于开发基于该接口的应用程序。

9.3.2 初始化

注意事项：

- config_path 配置文件可以是绝对路径，也可以是相对路径。如果是相对路径，先查找系统环境变量 CSP_BASE_PATH 指定的目录；然后查找下面 base_dir 指定的基础目录，最后会在程序根目录查找
- dll_path 可设置 DLL 库的绝对路径地址或者 DLL 库的目录地址。如果是相对路径，相对当前的目录路径
- base_dir 可以指定基础目录路径，其他相对路径都可以基于这个基础路径，如果不设置，默认设为 dll_path 的目录路径
- start 方法主要用于启动后台泵线程，已在初始化中自动调用，不需要显式调用

【API 定义】

API 函数	参数	返回值	功能描述	备注
__init__	config_path：配置文件路径 dll_path: DLL 库路径 base_dir:基础目录路径 create_timeout：创建超时(秒)	INST 实例	初始化 ChronosInst 实例	<create_timeout> 创建超时，默认 5 秒
start	-	-	启动后台泵线程	内部调用
stop	-	-	停止后台泵线程并释放资源	

9.3.3 设备连接

注意事项：

- 可以先调用 get_devices 获取设备后，再进行连接
- 操作设备前，必须先连接设备
- 操作结束后，不再操作设备，请先断开设备连接

【设备接口类型定义】

定义	值	说明

ConnType.USB3_DEV	1	Usb3.0 接口
ConnType.UDP_DEV	2	千兆网网口
ConnType.UDP_W_DEV	3	万兆网网口

【API 定义】

API 函数	参数	返回值	功能描述	说明
get_devices	-	List[DeviceInfo]	获取可用设备列表	设备信息包含类型和名称
connect	ConnType device_type：设备接口类型	bool	连接指定类型设备	具体类型定义请详见 ConnType 定义。
disconnect	-	bool	断开当前设备连接	

9.3.4 错误处理

【错误码定义】

定义	值	说明
Error.CONFIG_NOT_FOUND	2000	配置文件不存在
Error.INST_DEVICE_NOT_FOUND	2001	设备未找到
Error.INST_DEVICE_NOT_CONNECTED	2002	设备未连接
Error.INST_DEVICE_ALREADY_RUNNING	2003	设置已运行
Error.INST_START_PROTOCOL_FRAMEWORK	2004	启动协议层失败
Error.INST_PROTOCOL_FRAMEWORK_NOT_RUNNING	2005	协议层未运行
Error.INST_PROTOCOL_FRAMEWORK_INTERNAL	2006	协议层内部错误
Error.INST_INVALID_ACQ_TYPE	2007	错误的数据采集方式
Error.INST_ACQ_DATA_FAILED	2008	数据采集失败
Error.INST_PROCESS_ANALYSIS_FAILED	2009	数据分析失败
Error.INST_DATA_PROCESS_NOT_STOPPED	2010	上一次数据分析任务未完成
Error.INST_INVALID_CALIBRATION_PARAM	2011	错误的标定参数
Error.INST_CALIBRATION_FAILED	2012	设备标定失败
Error.INST_CONVERT_BINFILE_FAILED	2013	转换 BIN 文件失败

Error.INST_INVALID_PARAMETER	2014	错误的参数
Error.INST_TERMINATION_CONFIG_UNSUPPORTED	2015	不支持阻抗参数配置
Error.INST_ENABLE_CHANNEL_FAILED	2016	通道使能失败
Error.INST_DISABLE_CHANNEL_FAILED	2017	通道禁能失败
Error.INST_SET_CHANNELS_ENABLED_FAILED	2018	设置通道使能状态失败

【API 定义】

API 函数	参数	返回值	功能描述	备注
on_error	cb：错误回调函数	-	注册错误回调	回调格式:(Error, message)
get_last_error	-	(code, message)	获取最近错误信息	

9.3.5 设备自检

设备使用前，可以先进行设备自检操作，设置各通道和全局配置的初始默认值。设备自检后，在 selfcheck 目录下会生成自检日志文档，可以查看自检状态。

【API 定义】

API 函数	参数	返回值	功能描述	备注
self_check	-	bool	设备自检	

9.3.6 通道配置

注意事项：

- 参考通道不可设置 DAC、Termination、Hysteresis 和 A/D Integration Point 值
- 仅部分型号支持 Termination 阻抗配置（Venus Ultra 系列）
- A/D Integration Point 值，仅对 A/D 通道有效（设备的奇数通道）

【阻抗类型定义】

定义	值	说明

TermType.TERM_50	0	阻抗 50 Ohm（默认值）
TermType.TERM_1M	1	阻抗 1M Ohm

【迟滞电压类型定义】

定义	值	说明
HtsType.HTS_30_MV	0	迟滞电压 30mV（默认值）
HtsType.HTS_40_MV	1	迟滞电压 40mV
HtsType.HTS_70_MV	2	迟滞电压 70mV
HtsType.HTS_1_MV	3	迟滞电压 1mV

【边沿触发类型定义】

定义	值	说明
EdgeType.RISING_EDGE	0	上升沿
EdgeType.FALLING_EDGE	1	下降沿
EdgeType.PMIDDLE	2	正中间
EdgeType.NMIDDLE	3	负中间

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
set_dac	channel：通道号 value: DAC 值	bool	设置通道 DAC 电压	-1500~1500 mV
set_termination	channel：通道号 value：阻抗类型值	bool	设置通道阻抗类型值	详见前面的 TermType 定义
set_hysteresis	channel：通道号 value：迟滞电压类型值	bool	设置通道迟滞电压类型值	详见前面的 HtsType 定义
set_inter_delay	channel：通道号 value：通道延迟值	bool	设置通道间延迟值	0~1000000ps
set_dead_time	channel：通道号 value：死时间	bool	设置通道死时间	2~130000ps

set_edge_type	channel：通道号 value：边沿触发类型	bool	设置边沿触发类型	详见前面的 EdgeType 定义
set_ad_integration_point	channel：通道号 value：积分点	bool	设置 A/D 面积积分值	0~65535
enable_channel	channel：通道号	bool	设置通道使能	
disable_channel	channel：通道号	bool	设置通道禁能	
enable_all_channels	-	bool	设置所有通道使能	
disable_all_channels	-	bool	设置所有通道禁能	
get_dac	channel：通道号	float	获取通道 DAC 值	
get_hysteresis	channel：通道号	int	获取通道迟滞电压类型值	
get_inter_delay	channel：通道号	int	获取通道间延迟值	
get_dead_time	channel：通道号	int	获取通道死时间	
get_edge_type	channel：通道号	int	获取边沿触发类型	
get_ad_integration_point	channel：通道号	int	获取 A/D 面积积分值	
is_channel_enabled	channel：通道号	bool	获取通道使能状态	

9.3.7 全局配置

注意事项：

- 测试模式类型主要用于仿真和测试，正常使用采用 RAW 模式；
- 切换 TDC 时间，设备会重新进行校准，耗时较长（一般在 20s 左右）；
- 系统复位操作会重置设备所有设置，耗时较长（一般在 10s 左右），请谨慎操作；
- 板卡 ID 设置主要用于多设备并联的场景，用于区分不同设备。

【TDC 时钟来源类型定义】

定义	值	说明
ClockSource.INTERNAL_CLOCK	0	内部时钟
ClockSource.EXTERNAL_CLOCK	1	外部时钟

【数据模式定义】

定义	值	说明
DataMode.TIMESTAMP	0	时间模式
DataMode.TIME ZONE	1	时区模式
DataMode.AD	2	A/D 模式
DataMode.AREA	3	面积测量模式
DataMode.REF_COINS	4	参考符合模式
DataMode.GLOBAL_COINS	5	全局符合模式
DataMode.TOT	6	过阈时间测量模式
DataMode.AREA_COINS	7	能谱-全局符合模式
DataMode.DUAL_TIMESTAMP	8	双沿时间模式
DataMode.PERIOD	9	周期测量模式
DataMode.SINGLE_COINS	11	单符合模式
DataMode.GLOBAL_COINS_TIMESTAMP	12	全局符合（带时间戳）模式

【测试模式定义】

定义	值	说明
TestMode.RAW	0	RAW 原始数据模式
TestMode.ALL_0	1	全 0 模式
TestMode.ALL_1	2	全 1 模式
TestMode.TOGGLE	3	转换模式
TestMode.INCREASE	4	递增模式
TestMode.DECREASE	5	递减模式
TestMode.TRIGGER_1MHZ	6	1MHz 内部触发模式

【数据带宽类型定义】

定义	值	说明
TestDataBandwidth.DATA_BANDWIDTH_100M	0	100M 带宽
TestDataBandwidth.DATA_BANDWIDTH_1000M	1	1000M 带宽
TestDataBandwidth.DATA_BANDWIDTH_3000M	2	3000M 带宽
TestDataBandwidth.DATA_BANDWIDTH_6400M	3	6400M 带宽

【A/D 面积积分缩放类型定义】

定义	值	说明
ADAreaScale.SCALE_1	0	1 倍
ADAreaScale.SCALE_1_2	1	1/2 倍
ADAreaScale.SCALE_1_4	2	1/4 倍
ADAreaScale.SCALE_1_8	3	1/8 倍
ADAreaScale.SCALE_1_16	4	1/16 倍
ADAreaScale.SCALE_1_32	5	1/32 倍
ADAreaScale.SCALE_1_64	6	1/64 倍
ADAreaScale.SCALE_1_128	7	1/128 倍

【数据采集方式定义】

定义	值	说明
ACQType.ACQ_SOFTWARE	0	软件方式
ACQType.ACQ_HARDWARE	1	硬件方式

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
set_tdc_clock_source	src_type: 时钟源类型	bool	设置 TDC 时钟源	详见 ClockSource 定义
set_data_mode	data_mode: 数据模式	bool	设置数据模式	详见 DataMode 定义

set_test_mode	test_mode: 测试模式	bool	设置测试模式	详见 TestMode 定义
set_test_data_bandwidth	bw_type: 带宽类型	bool	设置测试数据带宽	详见 TestDataBandwidth 定义
set_board_id	board_id: 板卡 ID	bool	设置新板卡 ID	0~255
set_coins_time_window	value: 时间窗口	bool	设置符合时间窗口	0~16000000ps
set_ad_area_scale	value: 缩放类型	bool	设置 AD 区域缩放	详见 ADAreaScale 定义
set_acq_type	acq_type: 数据采集方式	bool	设置数据采集方式	详见 ACQType 定义
tdc_resetn	-	bool	TDC 复位	
system_resetn	-	bool	系统复位	
get_tdc_clock_source	-	int	获取 TDC 时钟源	
get_data_mode	-	int	获取数据模式	
get_test_mode	-	int	获取测试模式	
get_test_data_bandwidth	-	int	获取测试数据带宽	
get_board_id	-	int	获取板卡 ID	
get_coins_time_window	-	int	获取符合时间窗口	
get_ad_area_scale	-	int	获取 AD 区域缩放	
get_acq_type	-	int	获取数据采集方式	

9.3.8 状态查询

【设备状态定义】

定义	类型	说明
ResultDeviceStatus.temperature	int	设备核心温度
ResultDeviceStatus.current_1	float	电路 1 电流（mA）

ResultDeviceStatus.current_2	float	电路 2 电流（mA）
ResultDeviceStatus.current_3	float	电路 3 电流（mA）
ResultDeviceStatus.current_4	float	电路 4 电流（mA）
ResultDeviceStatus.current_5	float	电路 5 电流（mA）
ResultDeviceStatus.current_6	float	电路 6 电流（mA）
ResultDeviceStatus.current_7	float	电路 7 电流（mA）
ResultDeviceStatus.current_8	float	电路 8 电流（mA）

【API 定义】

API 函数	参数	返回值	功能描述	注意事项
get_channel_num	-	int	获取通道数量	获取设备通道数，不包含参考通道
get_device_type	-	int	获取设备类型	0x1： Venus(TDC) 0x2: Mercury(多道分析仪)
get_device_mode	-	int	获取设备型号	0x1: Venus Lite-32/Mercury-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0x10: Venus Lite 4 0x11: Venus Lite 2
get_product_sn	-	str	获取产品序列号	
get_device_status	-	ResultDeviceStatus	获取设备状态	包含温度和各电路电流，详见 ResultDeviceStatus 定

				义
get_device_temp	-	int	获取设备核心温度	
get_fan_speed	-	int	获取风扇转速	单位：RPM
get_fw_version	-	str	获取固件版本号	
get_sw_version	-	str	获取软件版本号	

9.3.9 网络配置

注意事项：

- 本机 IP 和设备 IP 需要在同一个网段
- 注意本地端口的占用情况，请不要设置已被占用的端口号
- 修改 IP 地址和网络端口后，请重连设备进行操作

【API 定义】

API 函数	参数	返回值	功能描述
set_ips	local_ip：本地 IP device_ip：设备 IP	bool	设置千兆网口 IP 地址
set_net_ports	local_port：本地端口 device_port：设备端口	bool	设置千兆网口端口号
set_wips	local_ip：本地 IP device_ip：设备 IP	bool	设置万兆网口 IP 地址
set_wnet_ports	local_port：本地端口 device_port：设备端口	bool	设置万兆网口端口号
get_ips	-	ResultIp	获取千兆网口 IP 地址
get_net_ports	-	ResultNetPort	获取千兆网口端口号
get_wips	-	ResultIp	获取万兆网口 IP 地址
get_wnet_ports	-	ResultNetPort	获取万兆网口端口号
set_mac	device_mac:网口 MAC 地址	bool	设置千兆网口 MAC 地址
get_mac	-	str	获取千兆网口 MAC 地址
set_wmac	device_mac:网口 MAC 地	bool	设置万兆网口 MAC 地址

	址		
get_wmac	-	str	获取万兆网口 MAC 地址

9.3.10 数据采集

注意事项：

- 数据采集 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- API 返回失败，可以在错误回调中接收到具体的错误信息，或调用 get_last_error 方法查看错误代码和错误信息；
- 提供了实时获取当前数据的接口，可以读取数据到内存进行实时处理；
- 可以调用 stop_acq API 去强制停止当前的数据采集任务；如果不调用，采集时间到了后，系统会自动调用 stop_acq API，停止数据采集；
- 启用十六进制格式输出，写文件速率较慢（默认二进制数据格式，文件写入更高效），耗时较长，请注意数据文件大小。

【API 定义】

API 函数	参数	返回值	说明
enable_hex_dataformat	-	bool	启用十六进制数据格式
disable_hex_dataformat	-	bool	禁用十六进制数据格式
is_hex_dataformat	-	bool	是否启用十六进制数据格式
set_rawdata_cache	buffer_size_mb	bool	设置 RAW 数据缓存，支持实时读取 设置缓存大小，单位：MB
acq_rawdata	data_mode：数据模式 duration：持续时间 file_path：数据文件目录（不设置，默认是 output 目录） file_name：数据文件名 max_volumesize_mb：文件分卷大小（默认为-1） force_close：是否强制停止文件保存（默认为 False，等待内存中数据全部保存）	bool	开始进行数据采集（异步模式） <data_mode> 请详见前面的 DataMode 定义 <duration> 采集时间：单位-秒 <file_name> 未明确后缀名情况下，如果启用十六进制格式输出，后缀名

			为“.dat”;如果未启用，默认是二进制格式输出，后缀名为“.bin” <max_volumesize_mb> 如果分卷大小小于等于 0，文件不问卷;
fetch_rawdata	timeout_ms：超时时间（单位：ms）	bytes	实时读取 Raw 数据 返回 byte 数组
fetch_rawdata_hex	timeout_ms：超时时间（单位：ms）	list[str]	实时读取 Raw 数据 返回十六进制字符串数组 （按 8 个字节分组）
fetch_rawdata_hex_string	timeout_ms：超时时间（单位：ms）	str	实时读取 Raw 数据 返回十六进制字符串 （按 8 个字节加上“\n”分隔）
fetch_rawdata_list	timeout_ms：超时时间（单位：ms）	list[int]	实时读取 Raw 数据 返回 int list 主要给 matlab 调用
stop_acq	-	bool	停止数据采集
convert_to_hex	bin_file：二进制原始数据文件路径 hex_file:输出的十六进制文件路径	bool	如果 hex_file 输入为空，默认生成的十六进制文件在二进制文件同级目录下

9.3.11 数据分析

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或调用 get_last_error 方法查看错误代码和错误信息；
- 可以调用 stop API 去强制停止当前的分析任务；如果不调用，分析时间到了后，系统会自动调用 stop API，停止分析。

【稳定性指标类型定义】

定义	值	说明
StabilityMetricType.ADEV	0	Allan Deviation (ADEV)
StabilityMetricType.MDEV	1	Modified Allan Deviation (MDEV)
StabilityMetricType.TDEV	2	Time Deviation (TDEV)
StabilityMetricType.HDEV	3	Hadamard Deviation (HDEV)

【API 定义】

API 函数	参数	返回值	功能描述	说明
get_single_cps	channel:通道号	int	获取通道实时计数率	
get_coins_cps	channel:通道号	int	获取通道符合计数率	必须先设置 DATA MODE 为 Global Coins (全局符合)
output_single_cps	duration：持续时间 folder_path：文件夹路径 file_name：文件名	bool	保存通道实时计数率到文件	<duration> 采集时间：单位-秒 <file_path> 未设置情况下，默认是 output 目录 <file_name> 未明确后缀名情况下，默认后缀名为 “.csv”
stop_singlecps_recording	-	bool	停止保存实时计数率	
output_coins_cps	duration：持续时间 folder_path：文件夹路	bool	保存通道符合计数率到	<duration> 采集时间：单位-

	径 file_name：文件名		文件	秒 <file_path> 未设置情况下，默认是 output 目录 <file_name> 未明确后缀名情况下，默认后缀名为“.csv”
stop_coinscps_recording	-	bool	停止保存符合计数率	
tof_analysis	duration：持续时间 ch1：通道 1 通道号 ch2：通道 2 通道号 coins_time_window：符合时间窗值（ps） avg_times:开启平均触发模式后的平均次数 folder_path：文件夹路径 file_name：文件名	bool	双通道 TOF 时间分析	<folder_path> 如果不设置，默认是在 output 目录 <file_name> 如果不设置，不保存文件
enable_cross_correlation_analysis	virtual_twin_ps：虚拟时间窗值（ps） bin_width_ps：BIN 宽度（ps） dcr1：通道 1 的暗计数率 dcr2：通道 2 的暗计数率	bool	启用互关联函数分析	
fetch_tof_processdata	-	List[int]	实时读取 TOF 分析数据	返回扁平 int 数组，格式为：<tdiff, counts>
fetch_cross_correlation_data	-	List[float]	实时读取互关联分析数据	返回扁平 float 数组，格式为：<tdiff, g2>
stop_tof_analysis	-	bool	停止 TOF 分析	
startstop_analysis	duration：持续时间 chan_groups：通道分	bool	开始终止通道时间差分	<chan_groups> Dict[int,

	组
	dual_stops_groups：双终止通道分组
	coins_time_window：符合时间窗值
	folder_path：文件夹路径
	file_name：文件名
			析	List[int]]
				示例：
				chan_groups={
				0: [1, 3, 5],
				# 开始通道 0：终止通道 1,3,5
				2: [4, 6]
				# 开始通道 2：终止通道 4，6
				}
				<dual_stops_groups>
				Optional[Dict[int, Tuple[int, int]]]
				示例：
				dual_stops_groups={
				0：(1，5)，# 开始通道 0：X-终止通道 1, Y-终止通道 5
				2：(4，6)
				# 开始通道 2：X-终止通道 4, Y-终止通道 6
				}

fetch_startstop_processdata	start_chan：开始通道号 stop_chan：终止通道号	List[int]	实时读取 Start-stop Process 数据	返回扁平 int 数组，格式为：<tdiff, counts>
fetch_startstop_average_processdata	start_chan：开始通道号	List[int]	实时读取 Start-stop Average Process 数据	返回扁平 int 数组，格式为：tdiff, counts
fetch_startstop_dualstops_rawdata	start_chan：开始通道号 timeout_ms：超时时间（单位：ms）	List[int]	实时读取 Period Process 数据	返回扁平 int 数组，格式为：<x_stop_tdiff, y_stop_tdiff>

stop_startstop_analysis	-	bool	停止开始终止通道时间差分析	
ncoins_analysis	duration: int, duration：持续时间 ncoins_groups：符合计数分析通道分组集合 coins_time_window：符合时间窗值 folder_path：文件夹路径 file_name：文件名	bool	开始 N 重符合计数分析	<ncoins_groups> List[Tuple[List[int], int]] 示例： ncoins_groups =[([0, 1, 2], 1000), # 通道[0, 1, 2], 虚拟时间窗 10000 ([3, 4], 1000), # 通道[3, 4], 虚拟时间窗 10000]
fetch_ncoins_rawdata	ncoins_channels：符合计数分析的通道组 timeout_ms：超时时间 （单位：ms）	List[int]	实时读取指定通道分组的 N 重符合计数分析的过程符合数据	返回扁平 int 数组，格式为：<coins_index, start_ch, start_timestamp,end_ch, end_timestamp>

fetch_ncoins_cps	ncoins_channels：符合计数分析的通道组	List[int]	实时读取指定通道分组的计数率 CPS 数据	返回扁平 int 数组，格式为： <time, cps>
stop_ncoins_analysis	-	bool	停止 N 重符合计数分析	
autocorrelation_analysis	duration：持续时间 ch：通道 min_tau：最小 tau(单位：s) max_tau：最大 tau（单位：s） bin_num：BIN 数量 edge_type：分析边沿类型（上升沿或下降沿） folder_path：文件夹路径 file_name：文件名	bool	单通道自关联函数（G2）分析	
fetch_autocorrelation_data	-	List[float]	实时读取自关联分析数据	返回扁平 float 数组，格式为：<tdiff, g2>
stop_autocorrelation_analysis	-	bool	停止自关联分析	
tot_analysis	duration：持续时间 ch：通道 folder_path：文件夹路径 file_name：文件名	bool	过阈值时间分析	
fetch_tot_processdata	-	List[int]	实时读取 Tot Process 数据	返回扁平 int 数组，格式为：<tot, counts>
stop_tot_analysis	-	bool	停止 TOT 分析	
period_analysis	duration：持续时间	bool	频率分析	<sampling_interval>

	decades：数据量级大小 decade_points：每个数据量的点数 average_num：平均计算次数 analysis_length：分析数据长度 fnom_hz：标称频率（Hz） sampling_interval：抽样间隔 sampling_num：抽样数量 folder_path：文件夹路径 file_name：文件名			周期抽样间隔，单位：ms <sampling_num> 周期抽样数量 如果小于等于 0，默认值为 100 数据抽样只是为了周期数据的展示，或者定量分析。
fetch_period_rawdata	timeout_ms：超时时间（单位：ms）	List[int]	实时读取周期抽样数据	返回扁平 int 数组，格式为：<index, period>
fetch_period_processdata	-	List[int]	实时读取周期统计分析数据	返回扁平 int 数组，格式为：<period, counts>
fetch_period_stability_metric	metric_type：稳定性指标类型（详见 StabilityMetricType 定义）	List[float]	实时读取稳定性指标数据	返回扁平 int 数组，格式为：<tau, value>
stop_period_analysis	-	bool	停止频率分析	
ad_analysis	duration：持续时间 ch：通道号	bool	A/D 信号分析	<ch> 必须是 A/D 通道

	ad_integration_point: A/D 面积积分值 sampling_interval：采样间隔 sampling_num：采样数量 folder_path：文件夹路径 file_name：文件名			<sampling_interval> 采样间隔，单位：ms <sampling_num> 如果小于等于 0，全采样
fetch_ad_rawdata	timeout_ms：超时时间（单位：ms）	List[int]	实时读取 A/D RAW 数据	返回扁平 int 数组，格式为：sampling_points, adc
stop_ad_analysis	-	bool	停止 A/D 信号分析	
es_analysis	duration：持续时间 ch：通道 ad_integration_point: A/D 面积积分值 ad_area_scale: A/D 面积积分缩放类型 folder_path：文件夹路径 file_name：文件名	bool	能谱分析	<ch> 必须是 A/D 通道
fetch_es_processdata	-	List[int]	实时读取 Tot Process 数据	返回扁平 int 数组，格式为：<adc, counts>
stop_es_analysis	-	bool	停止能谱分析	
conform_es_analysis	duration：持续时间 ch1：通道 1 ch2：通道 2 ad_integration_point: A/D 面积积分值 ad_area_scale: A/D 面积积分缩放类型 energy_window_min1:	bool	符合能谱分析	<ch1> 必须是 A/D 通道 <ch2> 必须是 A/D 通道 <energy_window_min1> <energy_window_max1>

	能量窗口最小值 1 energy_window_max1:能量窗口最大值 1 energy_window_min2:能量窗口最小值 2 energy_window_max2:能量窗口最大值 2 coins_time_window:符合时间窗值 avg_times:开启平均触发模式后的平均次数 folder_path：文件夹路径 file_name：文件名			<energy_window_min2> <energy_window_max2> 能量窗的有效值范围：0-16384
fetch_conform_es_processdata	-	List[int]	实时读取符合能谱分析数据	返回扁平 int 数组，格式为：<tdiff, counts>
fetch_conform_es_area1_processdata	-	List[int]	实时读取符合能谱 Area1 分析数据	返回扁平 int 数组，格式为：<adc, counts>
fetch_conform_es_area2_processdata	-	List[int]	实时读取符合能谱 Area2 分析数据	返回扁平 int 数组，格式为：<adc, counts>
stop_conform_es_analysis	-	bool	停止符合能谱分析	
clear_analysis_data	-	bool	清空数据分析缓存	

9.3.12 注意事项

1) 返回值结构：

- ResultIp: (local_ip, device_ip) 包含本地和设备 IP 地址；
- ResultNetPort: (local_port, device_port) 包含本地和设备端口号 。

2) 状态码：

定义	值	说明
CI_SUCCESS	0	操作成功
CI_FAIL	-1	操作失败
CI_TRUE	1	布尔值 True
CI_FALSE	0	布尔值 False

3) **线程安全：**

- 所有 API 调用都是线程安全的；
- 数据采集和数据分析操作是异步的；
- 每次数据采集和分析完，建议手动调用 stop API。

4) **版本支持**

支持 Python3.7.0 及以上版本

9.3.13 示例代码

典型的应用代码示例如下：

```
from chronos_inst import ChronosInst, Status, ErrorCode, ConnType, ClockSource, EdgeType,
DataMode, TestMode, TestDataBandwidth, ADAreaScale
import time
try:
    # ensure config path is absolute or exists; ChronosInst will throw error if not found
    tdc = ChronosInst("D:\\cinstapi\\config\\apiconfig.ini", "D:\\test")

    # register an error callback
    tdc.on_error(lambda e, msg: print("TDC ERROR:", e, msg))

    # SDK already started pump thread and created native instance
    devs = tdc.get_devices()
    print("devices:", devs)

    if not devs:
        raise RuntimeError("No device found")

    ok = tdc.connect(devs[0].type)
```

```
        print("connected:", ok)
        if not ok:
            raise RuntimeError("Try to connect first device failed")

    # process self-check
    res = tdc.self_check()
    print("self_check->:", res)
    ......

    # process channel config
    res = tdc.set_dac(0, 100)
    print("setDAC->:", res)
    ......

    # process global config
    res = tdc.set_data_mode(DataMode.DUAL_TIMESTAMP)
    print("set_data_mode->:", res)
    ......

    # get device info
    res = tdc.get_device_status()
    print("get_device_status->:", res)
    ......

    # process network config
    res = tdc.set_ips( “10.0.0.5” , “10.0.0.10” )
    print("set_ips->:", res)
    ......

    # acq raw data
    tdc.enable_hexfile_format()
    tdc.set_rawdata_cache(200)
    res = tdc.acq_rawdata(DataMode.TIMESTAMP, 10, "raw", "python-test")
    print("acq_rawdata->:", res)
    raw_bytes = tdc.fetch_rawdata(500)
    time.sleep(10)
```

```
    tdc.stop_acq()
    time.sleep(2)

    # process data analysis
    res = tdc.tof_analysis(10, 0, 1, 10000)
    print("tof_analysis ->", res)
    time.sleep(10)
    data = tdc.fetch_tof_processdata()
    tdc.stop_tof_analysis()
    time.sleep(2)
    res = tdc.tof_analysis(10, 0, 1, 10000, 3,"", "tof_test")
    print("tof_analysis ->", res)
    time.sleep(15)
    ...

    # tdc resetn
        ret = tdc.tdc_resetn()
        print("tdc_resetn ->", ret)

    # system resetn
        ret = tdc.system_resetn()
        print("system_resetn ->", ret)

        ok = tdc.disconnect()
        print("disconnected:", ok)

        tdc.stop()   # stop pump and clean up
except RuntimeError as e:
        print(f"TDC example error: {e}")
```

9.4 MATLAB API

9.4.1 概述

ChronosInst 是一个用于高精度测量设备（比如：TDC 时间数字转换器设备）

操作的 Matlab 封装类，提供设备连接、配置、数据采集和分析等功能。

这份文档涵盖了 ChronosInst Matlab API 的主要功能和使用方法，可用于开发基于该接口的应用程序。

9.4.2 初始化

注意事项：

- configPath 配置文件可以是绝对路径，也可以是相对路径。如果是相对路径，先查找系统环境变量 CSP_BASE_PATH 指定的目录；然后查找下面 base_dir 指定的基础目录，最后会在程序根目录查找
- dllPath 可设置 DLL 库的绝对路径地址或者 DLL 库的目录地址。如果是相对路径，相对当前的目录路径
- basDir 可以指定基础目录路径，其他相关路径都可以基于这个基础路径，如果不设置，默认设为 dllPath 的目录路径

【API 定义】

API 函数	参数	返回值	功能描述
ChronosInst(configPath, dllPath,baseDir)	configPath：配置文件路径 dllPath: DLL 库路径 baseDir:基础目录路径	INST 实例	初始化 ChronosInst 实例

9.4.3 设备连接

注意事项：

- 可以先调用 getDevices 获取设备后，再进行连接
- 操作设备前，必须先连接设备
- 操作结束后，不再操作设备，请先断开设备连接

【设备接口类型定义】

定义	值	说明
ConnType.USB3_DEV	1	Usb3.0 接口
ConnType.UDP_DEV	2	千兆网网口

ConnType.UDP_W_DEV	3	万兆网网口

【API 定义】

API 函数	参数	返回值	功能描述	说明
getDevices	-	[(type, name)]	获取可用设备列表	返回设备信息结构体数组。 设备信息结构体包含类型和名称
connect	connType：设备接口类型	logical	连接指定类型设备	具体类型定义请详见 ConnType 定义。
disconnect	-	logical	断开当前设备连接	

9.4.4 错误处理

注意事项：

- 错误码定义，是来着 Python API 返回的错误码（详见 Python Error 错误码定义）
- 如果 API 调用返回失败，可以调用 getLastError 方法去获取相关错误代码和错误信息

【API 定义】

API 函数	参数	返回值	功能描述	备注
getLastError	-	(errorCode, errorMsg)	获取最近错误信息	

9.4.5 设备自检

设备使用前，可以先进行设备自检操作，设置各通道和全局配置的初始默认值。设备自检后，在 selfcheck 目录下会生成自检日志文档，可以查看自检状态。

【API 定义】

API 函数	参数	返回值	功能描述	备注
selfCheck	-	logical	设备自检	

9.4.6 通道配置

注意事项：

- 参考通道不可设置 DAC、Termination、Hysteresis 和 A/D Integration Point 值
- 仅部分型号支持 Termination 阻抗配置（Venus Ultra 系列）
- A/D Integration Point 值，仅对 A/D 通道有效（设备的奇数通道）

【阻抗类型定义】

定义	值	说明
TermType.TERM_50	0	阻抗 50 Ohm（默认值）
TermType.TERM_1M	1	阻抗 1M Ohm

【迟滞电压类型定义】

定义	值	说明
HtsType.HTS_30_MV	0	迟滞电压 30mV（默认值）
HtsType.HTS_40_MV	1	迟滞电压 40mV
HtsType.HTS_70_MV	2	迟滞电压 70mV
HtsType.HTS_1_MV	3	迟滞电压 1mV

【边沿触发类型定义】

定义	值	说明
EdgeType.RISING_EDGE	0	上升沿
EdgeType.FALLING_EDGE	1	下降沿
EdgeType.PMIDDLE	2	正中间
EdgeType.NMIDDLE	3	负中间

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
setDAC	channel：通道号 value: DAC 值	logical	设置通道 DAC 电压	-1500~1500mV

setTermination	channel：通道号 value：阻抗类型值	logical	设置通道阻抗类型值	详见前面的 TermType 定义
setHysteresis	channel：通道号 value：迟滞电压类型值	logical	设置通道迟滞电压类型值	详见前面的 HtsType 定义
setInterDelay	channel：通道号 value：通道延迟值	logical	设置通道延迟值	0~1000000 ps
setDeadTime	channel：通道号 value：死时间	logical	设置通道死时间	2~130000ps
setEdgeType	channel：通道号 value：边沿触发类型	logical	设置边沿触发类型	详见前面的 EdgeType 定义
setADIntegrationPoint	channel：通道号 value：积分点	logical	设置 A/D 面积积分值	0~65535
enableChannel	channel：通道号	logical	设置通道使能	
disableChannel	channel：通道号	logical	设置通道禁能	
enableAllChannels	-	logical	设置所有通道使能	
disableAllChannels	-	logical	设置所有通道禁能	
getDAC	channel：通道号	double	获取通道 DAC 值	
getTermination	channel：通道号	int32	获取通道阻抗类型值	
getHysteresis	channel：通道号	int32	获取通道迟滞电压类型值	
getInterDelay	channel：通道号	int32	获取通道延迟值	
getDeadTime	channel：通道号	int32	获取通道死时间	
getEdgeType	channel：通道号	int32	获取边沿触发类型	
getADIntegrationPoint	channel：通道号	int32	获取 A/D 面积积分值	
isChannelEnabled	channel：通道号	logical	获取通道使能状态	

9.4.7 全局配置

注意事项：

- 测试模式类型主要用于仿真和测试，正常使用采用 RAW 模式
- 系统复位操作会重置设备所有设置，耗时较长（一般在 10s 左右），请谨慎操作
- 板卡 ID 设置主要用于多设备并联的场景，用于区分不同设备

【TDC 时钟来源类型定义】

定义	值	说明
ClockSource.INTERNAL_CLOCK	0	内部时钟
ClockSource.EXTERNAL_CLOCK	1	外部时钟

【数据模式定义】

定义	值	说明
DataMode.TIMESTAMP	0	时间模式
DataMode.TIME ZONE	1	时区模式
DataMode.AD	2	A/D 模式
DataMode.AREA	3	面积测量模式
DataMode.REF_COINS	4	参考符合模式
DataMode.GLOBAL_COINS	5	全局符合模式
DataMode.TOT	6	过阈时间测量模式
DataMode.AREA_COINS	7	能谱-全局符合模式
DataMode.DUAL_TIMESTAMP	8	双沿时间模式
DataMode.PERIOD	9	周期测量模式

【测试模式定义】

定义	值	说明
TestMode.RAW	0	RAW 原始数据模式
TestMode.ALL_0	1	全 0 模式
TestMode.ALL_1	2	全 1 模式

TestMode.TOGGLE	3	转换模式
TestMode.INCREASE	4	递增模式
TestMode.DECREASE	5	递减模式
TestMode.TRIGGER_1MHZ	6	1MHz 内部触发模式

【数据带宽类型定义】

定义	值	说明
TestDataBandwidth.S_100M	0	100M 带宽
TestDataBandwidth.S_1000M	1	1000M 带宽
TestDataBandwidth.S_3000M	2	3000M 带宽
TestDataBandwidth.S_6400M	3	6400M 带宽

【A/D 面积积分缩放类型定义】

定义	值	说明
ADAreaScale.SCALE_1	0	1 倍
ADAreaScale.SCALE_1_2	1	1/2 倍
ADAreaScale.SCALE_1_4	2	1/4 倍
ADAreaScale.SCALE_1_8	3	1/8 倍
ADAreaScale.SCALE_1_16	4	1/16 倍
ADAreaScale.SCALE_1_32	5	1/32 倍
ADAreaScale.SCALE_1_64	6	1/64 倍
ADAreaScale.SCALE_1_128	7	1/128 倍

【数据采集方式定义】

定义	值	说明
ACQType.ACQ_SOFTWARE	0	软件方式
ACQType.ACQ_HARDWARE	1	硬件方式

【API 定义】

API 函数	参数	返回值	功能描述	有效范围

setTdcClockSource	src_type: 时钟源类型	logical	设置 TDC 时钟源	详见 ClockSource 定义
setDataMode	data_mode: 数据模式	logical	设置数据模式	详见 DataMode 定义
setTestMode	test_mode: 测试模式	logical	设置测试模式	详见 TestMode 定义
setTestDataBandwidth	bw_type: 带宽类型	logical	设置测试数据带宽	详见 TestDataBandwidth 定义
setBoardId	board_id: 板卡 ID	logical	设置新板卡 ID	0~255
setCoinsTimeWindow	value: 时间窗口	logical	设置符合时间窗口	0~16000000ps
setADAreaScale	value: 缩放类型	logical	设置 A/D 积分缩放	详见 ADAreaScale 定义
setACQType	acqType:数据采集方式	logical	设置数据采集方式	详见 ACQType 定义
tdcResetn	-	logical	TDC 复位	
systemResetn	-	logical	系统复位	
getTdcClockSource	-	int32	获取 TDC 时钟源	
getDataMode	-	int32	获取数据模式	
getTestMode	-	int32	获取测试模式	
getTestDataBandwidth	-	int32	获取测试数据带宽	
getBoardId	-	int32	获取板卡 ID	
getCoinsTimeWindow	-	int32	获取符合时间窗口	
getADAreaScale	-	int32	获取 A/D 积分缩放	
getACQType	acqType:数据采集方式	logical	设置数据采集方式	

9.4.8 状态查询

【设备状态定义】

定义	类型	说明
temperature	int32	设备核心温度
current_1	double	电路 1 电流（mA）
current_2	double	电路 2 电流（mA）
current_3	double	电路 3 电流（mA）
current_4	double	电路 4 电流（mA）
current_5	double	电路 5 电流（mA）
current_6	double	电路 6 电流（mA）
current_7	double	电路 7 电流（mA）
current_8	double	电路 8 电流（mA）

【API 定义】

API 函数	参数	返回值	功能描述	注意事项
getChannelNum	-	int32	获取通道数量	获取设备通道数，不包含参考通道
getDeviceType	-	int32	获取设备类型	0x1： Venus(TDC) 0x2: Mercury(多道分析仪)
getDeviceMode	-	int32	获取设备型号	0x1: Venus Lite-32/Mercury-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0x10: Venus Lite 4 0x11: Venus Lite 2
getProductSN	-	char	获取产品序列号	

getDeviceStatus	-	(temperature, current_1, current_2, current_3, current_4, current_5, current_6, current_7, current_8)	获取设备状态	包含温度和各电路电流，详见前面设备状态定义
getDeviceTemp	-	Int32	返回设备核心温度	
getFanSpeed	-	Int32	返回设备风扇转速	单位：RPM
getFWVersion	-	char	返回固件版本号字符串	
getSWVersion	-	char	返回软件版本号字符串	

9.4.9 网络配置

注意事项：

- 本机 IP 和设备 IP 需要在同一个网段
- 注意本地端口的占用情况，请不要设置已被占用的端口号
- 修改 IP 地址和网络端口后，请重连设备进行操作

【API 定义】

API 函数	参数	返回值	功能描述
setIPs	local_ip：本地 IP device_ip：设备 IP	logical	设置千兆网口 IP 地址
setNetPorts	local_port：本地端口 device_port：设备端口	logical	设置千兆网口端口号
setWIPs	local_ip：本地 IP device_ip：设备 IP	logical	设置万兆网口 IP 地址
setWNetPorts	local_port：本地端口	logical	设置万兆网口端口号

	device_port：设备端口		
getIPs	-	(local_ip, device_ip)	获取千兆网口 IP 地址
getNetPorts	-	(local_port, device_port)	获取千兆网口端口号
getWIPs	-	(local_ip, device_ip)	获取万兆网口 IP 地址
getWNetPorts	-	(local_port, device_port)	获取万兆网口端口号
setMAC	device_mac:网口 MAC 地址	logical	设置千兆网口 MAC 地址
getMAC	-	char	获取千兆网口 MAC 地址
setMAC	device_mac:网口 MAC 地址	logical	设置万兆网口 MAC 地址
getMAC	-	char	获取万兆网口 MAC 地址

9.4.10 数据采集

注意事项：

- 数据采集 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑
- API 返回失败，可以调用 getLastError 方法查看错误代码和错误信息
- 提供了实时获取当前数据的接口，可以读取数据到内存进行实时处理
- 可以调用 stopAcq API 去强制停止当前的数据采集任务；如果不调用，采集时间到了后，系统会自动调用 stopAcq API，停止数据采集
- 启用十六进制格式输出，写文件速率较慢（默认二进制数据格式，文件写入更高效），耗时较长，请注意数据文件大小

【API 定义】

API 函数	参数	返回值	说明
enableHexDataFormat	-	logical	启用十六进制数据格式
disableHexDataFormat	-	logical	禁用十六进制数据格式
isHexDataFormat	-	logical	是否启用十六进制数据格式

setRawDataCache	buffer_size_mb	logical	开启 RAW 数据缓存，支持实时读取 设置缓存大小，单位：MB
acqRawData	data_mode：数据模式 duration：持续时间 file_path：数据文件目录（不设置，默认是 output 目录） file_name：数据文件名 max_volumesize_mb：文件分卷大小（单位：MB，默认为-1，不分卷） force_close：是否强制停止文件保存（默认为 False，等待内存中数据全部保存）	logical	开始数据采集 <data_mode> 请详见前面的 DataMode 定义 <duration> 采集时间：单位-秒 <file_name> 未明确后缀名情况下，如果启用十六进制格式输出，后缀名为“.dat”;如果未启用，默认是二进制格式输出，后缀名为“.bin”
fetchRawData	timeout_ms：超时时间（单位：ms）	(success, rawData])	实时读取 RAW 数据 <rawData> 是 uint8([])数组
fetchRawDataHex	timeout_ms：超时时间（单位：ms）	(success, hexStrLines)	实时读取 RAW 数据 <hexStrLines> 十六进制字符串数组，按每 8 个字节分组 示例：[ffffffffffffffff, 00000ff0004c034a, ...]
stopAcq	-	logical	停止数据采集
convertToHex	bin_file：二进制原始数据文件路径 hex_file:输出的十六进制文件路径	logical	如果 hex_file 输入为空，默认生成的十六进制文件在二进制文件同级目录下

9.4.11 数据分析

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；

- 如果 API 返回失败，可以调用 getLastError 方法查看错误代码和错误信息；
- 可以调用 stop API 去强制停止当前的分析任务；如果不调用，分析时间到了后，系统会自动调用 stop API，停止分析。

【API 定义】

API 函数	参数	返回值	功能描述	说明
getSingleCPS	channel:通道号	int	获取通道实时计数率	
getCoinsCPS	channel:通道号	int	获取通道符合计数率	必须先设置 DATA MODE 为 Global Coins (全局符合)
outputSingleCPS	duration：持续时间 folder_path：文件夹路径 file_name：文件名	logical	保存通道实时计数率到文件	<duration> 采集时间：单位-秒 <file_path> 未设置情况下，默认是 output 目录 <file_name> 未明确后缀名情况下，默认后缀名为“.csv”
stopSingleCPSRecording	-	logical	停止保存实时计数率	
outputCoinsCPS	duration：持续时间 folder_path：文件夹路径 file_name：文件名	logical	保存通道符合计数率到文件	<duration> 采集时间：单位-秒 <file_path> 未设置情况下，默认是 output 目录 <file_name> 未明确后缀名情况下，默认后缀名为“.csv”
stopCoinsCPSRecording	-	logical	停止保存符合计数率	
tofAnalysis	duration：持续时间 ch1：通道 1 通道号	logical	双通道 TOF 时间分析	<folder_path> 如果不设置，默认是

	ch2：通道 2 通道号 coins_time_window:符合时间窗值 avg_times:开启平均触发模式后的平均次数 folder_path：文件夹路径 file_name：文件名			在 output 目录 <file_name> 如果不设置，不保存文件
enableCrossCorrelationAnalysis	virtual_twin_ps:虚拟时间窗（单位：ps） bin_width_ps：BIN 宽度（单位：ps） dcr1：通道 1 的暗计数率 dcr2：通道 2 的暗计数率	logical	开启互关联函数（G2）分析	
fetchTofProcessData	-	int array	实时读取 TOF 分析数据	返回扁平 int 数组，格式为：<tdiff, counts>
fetchCrossCorrelationData	-	double array	实时读取互关联分析数据	返回扁平 double 数组，格式为：<tdiff,g2>
stopTofAnalysis	-	logical	停止 TOF 分析	
startStopAnalysis	duration：持续时间 chan_groups：通道分组 dual_stops_groups:双终止通道分组 coins_time_window:符合时间窗值 folder_path：文件夹路径 file_name：文件名	logical	开始终止通道时间差分析	<chan_groups> <dual_stops_groups> 格式：containers.Map 或 struct array 示例 1： chan_groups = [struct('start_chan', 0, 'stop_chans', [1, 3, 5]); struct('start_chan', 2, 'stop_chans', [4, 6])];

				% 定义 dual_stops_groups
				dual_stops_groups = [
				struct('start_chan', 0, 'dual_stops', [1, 5]);
				struct('start_chan', 2, 'dual_stops', [4, 6])
				];
				
				示例 2:
				chan_groups = containers.Map('KeyType','int32', 'ValueType','any');
				chan_groups(0) = [1, 3, 5];
				chan_groups(2) = [4, 6];
				
				dual_stops_groups = containers.Map('KeyType','int32', 'ValueType','any');
				dual_stops_groups(0) = [1, 5];
				dual_stops_groups(2) = [4, 6];

fetchStartStopProcessData	start_chan：开始通道号 stop_chan：终止通道号	int array	实时读取指定开始-终止通道组的分析数据	返回扁平 int 数组，格式为：<tdiff, counts>
fetchStartStopAverag	start_chan：开始通道	int	实时读取指定	返回扁平 int 数组，

eProcessData	号	array	通道分组的平均分析数据	格式为：<tdiff, counts>
fetchStartStopDualStopsRawData	start_chan：开始通道号 timeout_ms：超时时间（单位：ms）	int array	实时读取指定分组的双终止 RAW 数据	返回扁平 int 数组，格式为：<x_stop_tdiff, y_stop_tdiff>
stopStartStopAnalysis	-	logical	停止开始终止通道时间差分析	

nCoinsAnalysis	duration：持续时间
	ncoins_groups：通道分组集合
	coins_time_window：符合时间窗值
	folder_path：文件夹路径
	file_name：文件名
		logical	开始 N 重符合计数分析	<ncoins_groups>
				格式：
				struct array[containers.Map]
				示例 1：
				ncoins_groups(1).channels = [0, 1, 2];ncoins_groups(1).virtualTwin = 10000;
				ncoins_groups(2).channels = [3, 4];ncoins_groups(2).virtualTwin = 10000;

fetchNCoinsRawData	channels：通道号数组	int array	实时读取指定分组的符合计	返回扁平 int 数组，格式为：

			数过程数据	<coins_index, start_ch, start_timestamp, end_ch, end_timestamp>
fetchNCoinsCPS	channels：通道号数组	int array	实时读取指定分组的符合计数率 CPS 数据	返回扁平 int 数组，格式为：<time(s), cps>
stopNCoinsAnalysis	-	logical	停止 N 重符合计数分析	
autoCorrelationAnalysis	duration：持续时间 ch：通道 min_tau：最小 tau(单位：s) max_tau：最大 tau（单位：s） bin_num：BIN 数量 edge_type：分析边沿类型（上升沿或下降沿） folder_path：文件夹路径 file_name：文件名	logical	单通道自关联函数（G2）分析	
fetchAutoCorrelationData	-	double array	实时读取自关联分析数据	返回扁平 double 数组，格式为：<tdiff(s), g2>
stopAutoCorrelationAnalysis	-	logical	停止自关联函数分析	
totAnalysis	duration：持续时间 ch：通道 folder_path：文件夹路径 file_name：文件名	logical	过阈值时间分析	
fetchTotProcessData	-	int array	实时读取 Tot Process 数据	返回扁平 int 数组，格式为：<tot, counts>

stopTotAnalysis	-	logical	停止 TOT 分析	
periodAnalysis	duration：持续时间 decades：数据量级大小 decade_points：每个数据量的点数 average_num：平均计算次数 analysis_length：分析数据长度 fnom_hz：标称频率（Hz） sampling_interval:抽样间隔 sampling_num：抽样数量 folder_path：文件夹路径 file_name：文件名	logical	频率分析	<sampling_interval> 周期抽样间隔，单位：ms <sampling_num> 周期抽样数量 如果小于等于 0，默认值为 100 数据抽样只是为了周期数据的展示，或者定量分析。
fetchPeriodRawData	timeout_ms：超时时间（单位：ms）	int array	实时读取周期抽样数据	返回扁平 int 数组，格式为：<index, period>
fetchPeriodProcessData	-	int array	实时读取周期统计分析数据	返回扁平 int 数组，格式为：<period, counts>
fetchPeriodStabilityMetric	metric_type：稳定性分析指标类型	double array	实时读取频率分析稳定性指标	返回扁平 double 数组，格式为：<tau, value>
stopPeriodAnalysis	-	logical	停止频率分析	

adAnalysis	duration：持续时间 ch：通道号 ad_integration_point：A/D 面积积分值 sampling_interval:采样间隔 sampling_num：采样数量 folder_path：文件夹路径 file_name：文件名	logical	A/D 信号分析	<ch> 必须是 A/D 通道 <sampling_interval> 采样间隔，单位：ms <sampling_num> 如果小于等于 0，全采样
fetchAdRawData	timeout_ms：超时时间（单位：ms）	int array	实时读取 A/D RAW 数据	返回扁平 int 数组，格式为：<sampling_points, adc>
stopAdAnalysis	-	logical	停止 A/D 信号分析	
esAnalysis	duration：持续时间 ch：通道 ad_integration_point：A/D 面积积分值 ad_area_scale: A/D 面积积分缩放类型 folder_path：文件夹路径 file_name：文件名	logical	能谱分析	<ch> 必须是 A/D 通道
fetchEsProcessData	-	int array	实时读取 ES 分析数据	返回扁平 int 数组，格式为：<adc, counts>
stopEsAnalysis	-	logical	停止能谱分析	
conformEsAnalysis	duration：持续时间 ch1：通道 1 ch2：通道 2 ad_integration_point：A/D 面积积分值 ad_area_scale: A/D 面积积分缩放类型	logical	符合能谱分析	<ch1> 必须是 A/D 通道 <ch2> 必须是 A/D 通道 <energy_window_min1>

	energy_window_min1：能量窗口最小值 1 energy_window_max1：能量窗口最大值 1 energy_window_min2：能量窗口最小值 2 energy_window_max2：能量窗口最大值 2 coins_time_window:符合时间窗值 avg_times:开启平均触发模式后的平均次数 folder_path：文件夹路径 file_name：文件名			<energy_window_max1> <energy_window_min2> <energy_window_max2> 能量窗的有效值范围：0-16384
fetchConformEsProcessData	-	int array	实时读取符合能谱分析数据	返回扁平 int 数组，格式为：<tdiff, counts>
fetchConformEsArea1ProcessData	-	int array	实时读取符合能谱 Area1 分析数据	返回扁平 int 数组，格式为：<adc, counts>
fetchConformEsArea2ProcessData	-	int array	实时读取符合能谱 Area2 分析数据	返回扁平 int 数组，格式为：<adc, counts>
stopConformEsAnalysis	-	logical	停止符合能谱分析	
clearAnalysisData	-	logical	清空数据分析缓存	

9.4.12 注意事项

1） **API 使用前准备：**

- 请将最新的 Python API 接口文件（chronos_inst.py）拷贝到当前 matlab 执行目录的 python 子目录下；
- 请先执行 setup_chronos_inst.m，设置 python 安装目录；
- 请特别注意 matlab 版本和 python 版本的匹配情况，必须指定当前使用 matlab 匹配的 Python 版本。目前 Python API 接口支持 Python3.8.10+以

上版本。

2) 状态码：

定义	值	说明
SUCCESS	0	设置成功
INVALID	1	配置值和返回值不相等
FAIL	2	设置失败

3) 线程安全：

- 所有 API 调用都是线程安全的；
- 数据采集和数据分析操作是异步的；
- 每次数据采集和分析完，建议手动调用 stop API；
- 建议注册 onCleanup 方法，确保实例能正常关闭（如果实例不正常关闭，再次运行可能会导致异常的情况）。

  示例：cleanUpObj = onCleanup(@() tdc.disconnect());

4) 版本支持

支持 Matlab2019b 及以上版本

9.4.13 示例代码

典型的应用代码示例如下：

```
% chronos_tdc_example.m
% ChronosInst Matlab 使用示例

function chronos_tdc_example()
    %% 清理环境
    clear; clc; close all;

    fprintf('ChronosInst Matlab Example\n');
    fprintf('================================\n\n');

    %% 1. 运行诊断检查
    fprintf('1. Running diagnostics...\n');
    ChronosInst.runDiagnostics();

    %% 2. 初始化 TDC
```

```
fprintf('\n2. Initializing TDC...\n');

% 配置文件路径（请根据实际情况修改）
currentDir = fileparts(mfilename('fullpath'));
configPath = fullfile(currentDir, '..', '..', 'config', 'apiconfig.ini');
dllPath = fullfile(currentDir, '..', '..');
baseDir = ''; % If empty string, default to dll folder

% 检查配置文件是否存在
if ~exist(configPath, 'file')
    fprintf('Warning: Config file not found: %s\n', configPath);
    fprintf('Please update the configPath variable with correct path\n');
    configPath = input('Enter config file path: ', 's');
end

% 检查 DLL 文件是否存在
if ~exist(dllPath, 'file')
    fprintf('Warning: CINST dll file not found: %s\n', dllPath);
    fprintf('Please update the dllPath variable with correct path\n');
    dllPath = input('Enter CINST dll file path: ', 's');
end

try
% 创建 TDC 实例
    tdc = ChronosInst(configPath, dllPath);

    % 注册 cleanup：确保实例被关闭
    cleanUpObj = onCleanup(@() tdc.disconnect());

    fprintf('TDC initialized successfully!\n');
catch ME
    fprintf('Failed to initialize TDC: %s\n', ME.message);
    fprintf('Please check:\n');
    fprintf('1. Run setup_chronos_tdc to configure Python environment\n');
    fprintf('2. Python is installed and accessible\n');
    fprintf('3. chronos_inst.py is in matlab script directory\n');
    fprintf('4. Config file path and dll file path is correct\n');
    fprintf('\nTip: Run setup_chronos_inst() for automatic setup\n');
    return;
end

%% 3. 获取设备列表
```

```
fprintf('\n3. Getting device list...\n');

try
    devices = tdc.getDevices();
    if isempty(devices)
        fprintf('No devices found\n');
    else
        fprintf('Found %d device(s):\n', length(devices));
        for i = 1:length(devices)
            fprintf(' Device %d: Type=%d, Name=''%s''\n', ...
            i, devices(i).type, devices(i).name);
        end
    end
catch ME
    fprintf('Failed to get devices: %s\n', ME.message);
end

%% 4. 连接设备（如果有设备的话）
if exist('devices', 'var') && ~isempty(devices)
    fprintf('\n4. Connecting to device...\n');
    try
        % 连接第一个设备
        deviceType = devices(1).type;
        success = tdc.connect(deviceType);
        if success
            fprintf('Connected to device successfully!\n');
            %% 5. 获取板卡 ID
            fprintf('\n5. Getting board ID...\n');
            boardId = tdc.getBoardId();
            fprintf('Board ID: %d\n', boardId);

            %% 6. 设备自检操作示例
            fprintf('\n6. Device self-check operations...\n');
            result = tdc.selfCheck();
            if result
            fprintf('Process self-check successful\n');
            else
            fprintf('Process self-check falied\n');
            end

            %% 7. 通道配置操作示例
            fprintf('\n8. Channel configuration operations...\n');
```

```
% 设置 DAC
channel = 0;
value = 100;
fprintf('Setting DAC channel %d to %.2fmV...\n', channel, value);
result = tdc.setDAC(channel, value);
if result
    fprintf('DAC set successfully: %.2fmV\n', value);
else
    fprintf('DAC set failed\n');
end
......

%% 8. 全局配置操作示例
fprintf('\n9. Global configuration control...\n');

% 设置时钟来源
fprintf('Setting clock source type
to %d...\n',int32(ClockSource.INTERNAL_CLOCK));
result = tdc.setTdcClockSource(ClockSource.INTERNAL_CLOCK);
if result
    fprintf('Clock source type set successfully\n');
else
    fprintf('Clock source type set failed\n');
End
......
%% 9. 设备相关数据获取示例
fprintf('\n10. Get device info...\n');

result = tdc.getChannelNum();
fprintf('Get channel num = %d\n', result);
......

%% 10. 网络设置示例
fprintf('\n11. Network configuration control...\n');
result = tdc.setIPs("10.0.0.5", "10.0.0.10");
result = tdc.getIPs();
fprintf('Get local ip = %s\n', result.local_ip);
fprintf('Get device ip = %s\n', result.device_ip);
......

%% 11. 采集控制示例
fprintf('\n12. Data acquisition control...\n');
```

```
% 设置输出 16 进制文件格式
tdc.enableHexFileFormat();

% 启用采集
fprintf('Starting acquisition...\n');
tdc.setTestMode(TestMode.RAW);
result = tdc.acqRawData(DataMode.TIMESTAMP, 10, "raw", "matlab-test");
if result
    fprintf('Acquisition started\n');
else
    fprintf('Failed to start acquisition\n');
end
% 等待一下
pause(10);
% 停止采集
fprintf('Stopping acquisition...\n');
result = tdc.stopAcq();

%% 12. 数据分析操作示例
fprintf('\n13. Data analysis control...\n');

% 强度分析
tdc.setDataMode(DataMode.TIMESTAMP);
tdc.setTestMode(TestMode.TRIGGER_10K);
result = tdc.outputSingleCPS(10, "", "");
if result
    fprintf('Output single cps data successful\n');
else
    fprintf('Failed to output single cps data\n');
end

% 等待分析时间
pause(10);

% 停止分析
result = tdc.stopSingleCPSRecording();
......

%% 14. 重置操作示例
fprintf('\n14. Resetn control...\n');
result = tdc.tdcResetn();
if result
```

```
                fprintf('TDC resetn successful\n');
            else
                fprintf('Failed to process TDC resetn\n');
            end

        else
            fprintf('Failed to connect to device\n');
        end
    catch ME
        fprintf('Error during device operations: %s\n', ME.message);
    end
else
    fprintf('\nNo devices available for connection test\n');
end
end
```

9.5 LabVIEW API

9.5.1 概述

ChronosInst 是一个用于高精度测量设备(比如:TDC 时间数字转换器设备)操作的 LabVIEW 封装类，提供设备连接、配置、数据采集和分析等功能。

这份文档涵盖了 ChronosInst LabVIEW Class 的主要功能和使用方法,可用于开发基于该功能类的应用程序。

9.5.2 环境准备

目前 ChronosInst LabVIEW 封装类是通过 Nexuslab 服务中转,和设备进行交互操作，请确保已提前安装 Nexuslab 服务。

- 启动 Nexuslab 服务，如果在同一台电脑设备，保持默认 IP 和端口即可;如果不同电脑设备，请确保网络可用，在配置文件中指定特定的 IP 和端口;
- 拷贝整个项目文件到特定目录，确保目录结构不变；确保 Config/apiconfig.ini 指向正确的 Nexuslab 服务;
- 确保 Lib/nexusapiXX.dll 文件存在，有 32 位/64 位两个 DLL 文件，对应特定的 LabVIEW 版本;

- 所有示例都在 Examples 目录下，请仔细查看；
- 支持 LabVIEW2010 及以后版本。

9.5.3 初始化和销毁

注意事项

- config path 配置文件可以是绝对路径，也可以是相对路径。如果是相对路径，基于 base dir 指定的基础目录查找；
- base dir 可以指定基础目录路径，其他相关路径都可以基于这个基础路径。

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciCreate	config path：配置文件路径 base path：基础目录路径 error in：错误输入	ChronosInst out：ChronosInst 实例 error out：错误输出	初始化 ChronosInst 实例	<base path> 默认指向当前工程的根目录 <config path> 默认指向当前工程的 Config/apiconfig.ini 文件
ciDestroy	ChronosInst in：ChronosInst 实例 error in：错误输入	error out：错误输出	销毁 ChronosInst 实例	

9.5.4 设备连接

注意事项：

- 先调用 ciGetDevices 获取设备后，再进行连接；
- 操作设备前，必须先连接设备；
- 操作结束后，不再操作设备，请先断开设备连接。

【设备接口类型定义】

定义	值	说明
CI_USB3_DEV	1	Usb3.0 接口

CI_UDP_DEV	2	千兆网网口
CI_UDP_W_DEV	3	万兆网网口

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciGetDevices	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 device num:设备数量 device list:设备列表（设备接口簇数组） device types:设备接口类型数组 device names:设备接口名称数组 error out：错误输出	获取当前可用的设备列表	
ciConnect	ChronosInst in: ChronosInst 实例 device type：设备接口类型值 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	连接设备	<ret> 状态码 0 -- 成功 -1 -- 失败
ciDisconnect	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	断开设备连接	

9.5.5 设备自检

设备使用前，可以先进行设备自检操作，设置各通道和全局配置的初始默认值。设备自检后，在 selfcheck 目录下会生成自检日志文档，可以查看自检状态。

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciSelfCheck	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设备自检	

9.5.6 通道配置

注意事项：

- 设备使用前，需要对各通道进行标定，获取各通道的标定值（可以在设备配套的 GUI 程序中进行设备标定操作）；
- 参考通道不可设置 DAC、Termination、Hysteresis 和 A/D Integration Point 值；
- 仅部分型号支持 Termination 阻抗参数配置（Venus Ultra 系列）；
- A/D Integration Point 值，仅对 A/D 通道有效（设备的奇数通道）。

【阻抗类型定义】

定义	值	说明
TERM_50	0	阻抗 50 Ohm（默认值）
TERM_1M	1	阻抗 1M Ohm

【迟滞电压类型定义】

定义	值	说明
HTS_30_MV	0	迟滞电压 30mV（默认值）
HTS_40_MV	1	迟滞电压 40mV
HTS_70_MV	2	迟滞电压 70mV
HTS_1_MV	3	迟滞电压 1mV

【边沿触发类型定义】

定义	值	说明
RISING_EDGE	0	上升沿

FALLING_EDGE	1	下降沿
PMIDDLE	2	正中间
NMIDDLE	3	负中间

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciSetDAC	ChronosInst in: ChronosInst 实例 channel：通道号 dac: DAC 设置值 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道的 DAC 值	DAC 值范围：-2000mV ~ 2000mV
ciSetTermination	ChronosInst in: ChronosInst 实例 channel：通道号 termination type：阻抗类型值 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道的阻抗类型值	详见前面的阻抗类型值定义（0-1）
ciSetHysteresis	ChronosInst in: ChronosInst 实例 channel：通道号 hysteresis type：迟滞电压类型值 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道的迟滞电压类型值	详见前面的迟滞电压类型值定义（0-3）
ciSetInterDelay	ChronosInst in: ChronosInst 实例 channel：通道号 inter delay：时间延迟值 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道时间延迟值	0~1000000ps
ciSetDeadTime	ChronosInst in: ChronosInst 实例	ChronosInst out: ChronosInst 实	设置通道死时间	2~130000ps

	channel：通道号 dead time：死时间 error in：错误输入	例 ret：操作状态码 error out：错误输出		
ciSetEdgeType	ChronosInst in: ChronosInst 实例 channel：通道号 edge type：边沿触发类型 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置边沿触发类型	详见前面的边沿触发类型定义（0-3）
ciSetADIntegrationPoint	ChronosInst in: ChronosInst 实例 channel：通道号 A/D integration point: A/D 积分点数 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置 A/D 面积积分值	范围：0~65535
ciEnableChannel	ChronosInst in: ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道使能	
ciDisableChannel	ChronosInst in: ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道禁能	
ciEnableAllChannels	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误	设置所有通道使能	

		输出		
ciDisableAllChannels	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置所有通道禁能	
ciGetDAC	ChronosInst in: ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 dac: DAC 设置值 error out：错误输出	获取通道 DAC 值	
ciGetHysteresis	ChronosInst in: ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 hysteresis type: 迟滞电压类型值 error out：错误输出	获取通道迟滞电压类型值	
ciGetInterDelay	ChronosInst in: ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 inter delay：时间延迟值 error out：错误输出	获取时间延迟值	
ciGetDeadTime	ChronosInst in: ChronosInst 实例 channel：通道号	ChronosInst out: ChronosInst 实例	获取通道死时间	

	error in：错误输入	ret：操作状态码 dead time：死时间 error out：错误输出		
ciGetEdgeType	ChronosInst in: ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 edge type：边沿触发类型 error out：错误输出	获取边沿触发类型	
ciGetADIntegrationPoint	ChronosInst in: ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 A/D integration point: A/D 积分点数 error out：错误输出	获取 A/D 面积积分值	
ciIsChannelEnabled	ChronosInst in: ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 is channel enabled：通道使能状态 error out：错误输出	获取通道使能状态	

9.5.7 全局配置

注意事项：

- 测试模式类型主要用于仿真和测试，正常使用采用 RAW 模式；
- 切换 TDC 时钟源，设备会重新进行校准，耗时较长（一般在 20s 左右）；
- 系统复位操作会重置设备所有设置，耗时较长（一般在 10s 左右），请谨慎操作；
- 板卡 ID 设置主要用于多设备并联的场景，用于区分不同设备。

【TDC 时钟来源类型定义】

定义	值	说明
INTERNAL_CLOCK	0	内部时钟
EXTERNAL_CLOCK	1	外部时钟

【数据模式定义】

定义	值	说明
DATA_MODE_TIMESTAMP	0	时间模式
DATA_MODE_TIME ZONE	1	时区模式
DATA_MODE_AD	2	A/D 模式
DATA_MODE_AREA	3	面积测量模式
DATA_MODE_REF_COINS	4	参考符合模式
DATA_MODE_GLOBAL_COINS	5	全局符合模式
DATA_MODE_TOT	6	过阈时间测量模式
DATA_MODE_AREA_COINS	7	能谱-全局符合模式
DATA_MODE_DUAL_TIMESTAMP	8	双沿时间模式
DATA_MODE_PERIOD	9	周期测量模式

【测试模式定义】

定义	值	说明
TEST_MODE_RAW	0	RAW 原始数据模式
TEST_MODE_ALL_0	1	全 0 模式
TEST_MODE_ALL_1	2	全 1 模式
TEST_MODE_TOGGLE	3	转换模式

TEST_MODE_INCREASE	4	递增模式
TEST_MODE_DECREASE	5	递减模式
TEST_MODE_TRIGGER_1MHZ	6	1MHz 内部触发模式

【数据带宽类型定义】

定义	值	说明
TEST_DATA_BANDWIDTH_100M	0	100M 带宽
TEST_DATA_BANDWIDTH_1000M	1	1000M 带宽
TEST_DATA_BANDWIDTH_3000M	2	3000M 带宽
TEST_DATA_BANDWIDTH_6400M	3	6400M 带宽

【A/D 面积积分缩放类型定义】

定义	值	说明
AREA_SCALE_1	0	1 倍
AREA_SCALE_1_2	1	1/2 倍
AREA_SCALE_1_4	2	1/4 倍
AREA_SCALE_1_8	3	1/8 倍
AREA_SCALE_1_16	4	1/16 倍
AREA_SCALE_1_32	5	1/32 倍
AREA_SCALE_1_64	6	1/64 倍
AREA_SCALE_1_128	7	1/128 倍

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciSetTdcClockSource	ChronosInst in: ChronosInst 实例 TDC clock source: TDC 时钟来源类型 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置 TDC 的时钟来源	详见前面的 TDC 时钟来源定义

ciSetDataMode	ChronosInst in: ChronosInst 实例 data mode：数据模式类型 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置数据模式	详见前面的 Data Mode 定义
ciSetTestMode	ChronosInst in: ChronosInst 实例 test mode：测试模式 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置测试模式	详见 TestMode 定义
ciSetTestDataBandwidth	ChronosInst in: ChronosInst 实例 test data bandwidth：测试数据带宽类型 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置测试数据带宽	详见 TestDataBandwidth 定义
ciSetBoardId	ChronosInst in: ChronosInst 实例 board id：板卡 ID error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置新板卡 ID	范围：0~255
ciSetCoinsTimeWindow	ChronosInst in: ChronosInst 实例 coins time window: 符合时间窗值 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码	设置符合时间窗值	范围：0~16000000ps

		error out：错误输出		
ciSetADAreaScale	ChronosInst in: ChronosInst 实例 A/D area scale: A/D 面积缩放类型 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置 AD 面积积分缩放类型	详见 ADAreaScale 定义
ciTdcResetn	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	TDC 复位	
ciSystemResetn	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	系统复位	系统复位后，需要等待一段时间再操作
ciGetDataMode	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 data mode：数据模式类型 error out：错误输出	获取数据模式	
ciGetTestMode	ChronosInst in: ChronosInst 实例	ChronosInst out:	获取测试模式	

	error in：错误输入	ChronosInst 实例 ret：操作状态码 test mode：测试模式 error out：错误输出		
ciGetTestDataBandwidth	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 test data bandwidth：测试数据带宽类型 error out：错误输出	获取测试数据带宽	
ciGetBoardId	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 board id：板卡 ID error out：错误输出	获取板卡 ID	如果失败，会返回-1
ciGetCoinsTimeWindow	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 coins time window：符合时间窗值 error out：错误输出	获取符合时间窗值	

ciGetADAreaScale	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 A/D area scale: A/D 面积缩放类型 error out：错误输出	获取 A/D 面积积分缩放类型	

9.5.8 状态查询

【设备状态簇】

定义	类型	说明
temperature	int	设备核心温度
current_1	float	电路 1 电流（mA）
current_2	float	电路 2 电流（mA）
current_3	float	电路 3 电流（mA）
current_4	float	电路 4 电流（mA）
current_5	float	电路 5 电流（mA）
current_6	float	电路 6 电流（mA）
current_7	float	电路 7 电流（mA）
current_8	float	电路 8 电流（mA）

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciGetChannelNum	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 channel num: 设备通道数 error out：错误	获取通道数量	获取设备通道数，不包含参考通道

		输出		
ciGetDeviceType	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 device type：设备类型 error out：错误输出	获取设备类型	0x1：Venus(TDC) 0x2: Mercury(多道分析仪)
ciGetDeviceMode	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 device mode: 设备型号 error out：错误输出	获取设备型号	0x1: Venus Lite-32/Mercury-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0xA: Venus lite4_2ps 0xB: Venus lite2_1ps
ciGetProductSN	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 product sn：设备产品序列号 error out：错误	获取产品序列号	

		输出		
ciGetDeviceStatus	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 device status: 设备状态 error out：错误输出	获取设备状态	包含温度和各电路电流，详见前面簇定义
ciGetDeviceTemp	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 device temp: 设备核心温度 error out：错误输出	获取设备核心温度	
ciGetFanSpeed	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 fan speed：设备风扇转速 error out：错误输出	获取设备风扇转速	单位：RPM

9.5.9 网络配置

注意事项：

- 本机 IP 和设备 IP 需要在同一个网段；
- 注意本地端口的占用情况，请不要设置已被占用的端口号；
- 修改 IP 地址和网络端口后，请重连设备进行操作；
- 修改 MAC 地址后，需要重启网络。

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciSetIPs	ChronosInst in: ChronosInst 实例 1G local IP：千兆网本地 IP 1G device IP：千兆网设备 IP error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置千兆网 IP	
ciSetNetPorts	ChronosInst in: ChronosInst 实例 1G local port：千兆网本地网络端口 1G device port：千兆网设备网络端口 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置千兆网口端口号	
ciSetWIPs	ChronosInst in: ChronosInst 实例 10G local IP：万兆网本地 IP 10G device IP：万兆网设备 IP error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置万兆网口 IP 地址	
ciSetWNetPorts	ChronosInst in: ChronosInst 实例 10G local port：万兆网本地网络端口 10G device port：万兆网设备网络端口 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置万兆网口端口号	
ciGetIPs	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 1G local IP：千兆网本地 IP 1G device IP：千兆网设备 IP	获取千兆网口 IP 地址	

		error out：错误输出		
ciGetNetPorts	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 1G local port：千兆网本地网络端口 1G device port: 千兆网设备网络端口 error out：错误输出	获取千兆网口端口号	
ciGetWNetPorts	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 10G local port: 万兆网本地网络端口 10G device port: 万兆网设备网络端口 error out：错误输出	获取万兆网口 IP 地址	
ciSetWNetPorts	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 10G local port: 万兆网本地网络端口 10G device port: 万兆网设备网络端口 error out：错误输出	获取万兆网口端口号	
ciSetMAC	ChronosInst in: ChronosInst 实例 1G device MAC：千兆网	ChronosInst out: ChronosInst 实例 ret：操作状态码	设置千兆网口 MAC 地址	

	设备 MAC 地址 error in：错误输入	error out：错误输出		
ciGetMAC	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 1G device MAC：千兆网设备 MAC 地址 error out：错误输出	获取千兆网口 MAC 地址	
ciSetWMAC	ChronosInst in：ChronosInst 实例 10G device MAC：万兆网设备 MAC 地址 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置万兆网口 MAC 地址	
ciGetWMAC	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 10G device MAC：万兆网设备 MAC 地址 error out：错误输出	获取万兆网口 MAC 地址	

9.5.10 数据采集

注意事项：

- 数据采集 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 提供了实时获取当前数据的接口，可以读取数据到内存进行实时处理；
- 可以调用 stop API 去强制停止当前的数据采集任务；如果不调用，采集时间到了后，系统会自动调用 stop API，停止数据采集；
- 启用十六进制格式输出，写文件速率较慢（默认二进制数据格式，文件写入更高效），耗时较长，请注意数据文件大小；
- 考虑到数据量问题，提供了采集数据后保存到远程服务端文件和本地文件的

不同接口，请根据数据量合理采用。

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciEnableHexDataFormat	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	启用十六进制数据格式	
ciDisableHexDataFormat	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	禁用十六进制数据格式	
ciIsHexDataFormat	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 Is hex data format：是否启用十六进制数据格式 error out：错误输出	是否启用十六进制数据格式	
ciSetRAWDataCache	ChronosInst in: ChronosInst 实例 buffer size(MB): 数据缓存大小 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	开启 RAW 数据缓存，支持实时读取	缓存大小，单位：MB
ciAcqRAWData	ChronosInst in: ChronosInst 实例 duration：数据采集持续时间 data mode：数据模式设置 file path：数据	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	开始进行数据采集（异步模式）	<data mode> 请详见前面的 DataMode 定义 <duration> 采集时间：单

	文件目录 file name：数据文件名 max volume size (MB)：文件分卷大小（默认为-1，不分卷） force close：是否强制停止文件保存（默认是 false，等待内存中数据全部保存） error in：错误输入			位-秒 <file path> 默认是 raw 目录 <file name> 未明确后缀名情况下，如果启用十六进制格式输出，后缀名为“.dat”;如果未启用，默认是二进制格式输出，后缀名为“.bin” <max volume size MB> 如果分卷大小小于等于 0，文件不分卷；
ciFetchRAWData	ChronosInst in：ChronosInst 实例 timeout(ms)：超时时间（单位：毫秒） error in：错误输入	ChronosInst out：ChronosInst 实例 data：U8 字节数组 error out：错误输出	实时读取 Raw 数据 返回 U8 字节数组	
ciStopAcq	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	停止数据采集	
ciConvertToHex	ChronosInst in：ChronosInst 实例 bin file path：BIN 文件路径 hex file path：输出的 HEX 文件路	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	在服务端转换 BIN 文件为 HEX 文件	如果 hex file path 输入为空，默认生成的十六进制文件在二进制文件同级目录下

	径 error in：错误输入			
ciConvertToHexLocal	ChronosInst in：ChronosInst 实例 bin file path：BIN 文件路径 hex file path：输出的 HEX 文件路径 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	本地转换 BIN 文件为 HEX 文件	
acqRAWDataToFile	ChronosInst in：ChronosInst 实例 duration：数据采集持续时间 duration delay：采集延迟时间 data mode：数据模式设置 file path：数据文件目录 file name：数据文件名 max volume size(MB)：文件分卷大小（默认为 0） force close：是否强制停止文件保存（false） error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	采集数据并保存到文件	duration delay 默认是 2 秒

9.5.11 数据分析

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 可以调用 stop API 去强制停止当前的分析任务；如果不调用，分析时间到了后，系统会自动调用 stop API，停止分析；
- 每个类型的分析，都提供了输出文件的接口，可以直接调用。

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciGetSingleCPS	ChronosInst in: ChronosInst 实例 channel：指定通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 single cps：实时计数率 error out：错误输出	获取指定通道的实时计数率值	
ciOutputSingleCPS	ChronosInst in: ChronosInst 实例 duration second: 数据采集持续时间 file path：数据文件目录 file name：数据文件名 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	输出所有通道的实时计数率到文件	<file path> 默认是 output 目录
ciStopSingleCPSRecording	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	停止记录实时计数率到文件	
ciGetCoinsCPS	ChronosInst in: ChronosInst 实例 channel：指定通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 coins cps：符合计数率 error out：错误输出	获取指定通道的符合计数率值	
ciOutputCoinsCPS	ChronosInst in: ChronosInst 实例 duration second: 数据采集持续时间	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	输出所有通道的符合计数率到文件	<file path> 默认是 output 目录

	file path：数据文件目录 file name：数据文件名 error in：错误输入			
ciStopCoinsCPSRecording	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	停止记录符合计数率到文件	
ciTofAnalysis	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 ch1：通道 1 的通道号 ch2：通道 2 的通道号 coins time window：符合时间窗值 avg times：平均触发模式下的次数 file path：数据文件目录 file name：数据文件名 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	TOF 数据分析并保存数据到文件 ● RAW 实时数据文件 ● Process 分析数据文件	<avg times> 大于 0，启用平均触发模式；小于等于 0，默认是单次触发模式 <file path> 默认是 output 目录
ciFetchTofProcessData	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 TOF Process 数据	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：tdiff, counts
ciStopTofAnalysis	ChronosInst in: ChronosInst 实例	ChronosInst out: ChronosInst 实例	停止 TOF 数据分析	

	error in：错误输入	ret：操作状态码 error out：错误输出		
tofAnalysisToFile	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 duration delay：分析延迟时间 ch1：通道 1 的通道号 ch2：通道 2 的通道号 coins time window：符合时间窗值 avg times：平均触发模式下的次数 file path：数据文件目录 file name：数据文件名 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	TOF 数据分析并保存数据到文件 ● RAW 实时数据文件 ● Process 分析数据文件	<duration delay> 默认 2 秒 <avg times> 大于 0，启用平均触发模式；小于等于 0，默认是单次触发模式 <file path> 默认是 output 目录
ciADAnalysis	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 ch：指定的通道号 A/D integration point: A/D 面积积分值 sampling interval(ms)：采样间隔时间（单位：毫秒） sampling number: 采样数量 file name：数据文件名	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	A/D 数据分析并保存数据到文件 ● RAW 实时数据文件	<ch> 必须指定 A/D 通道号 <file path> 默认是 output 目录

	error in：错误输入			
ciFetchADRAWData	ChronosInst in：ChronosInst 实例 timeout(ms)：超时时间（单位：毫秒） error in：错误输入	ChronosInst out：ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 A/D RAW 数据	1）在超时时间内，直到获取数据才返回； 2）返回扁平 Int64 数组，二个数字一组，格式为：sampling_points, adc
ciStopADAnalysis	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	停止 A/D 数据分析	
adAnalysisToFile	ChronosInst in：ChronosInst 实例 duration second：数据分析持续时间 duration delay：分析延迟时间 ch：指定的通道号 A/D integration point: A/D 面积积分值 sampling interval(ms)：采样间隔时间（单位：毫秒） sampling number：采样数量 file name：数据文件名 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	A/D 数据分析并保存数据到文件 ● RAW 实时数据文件	<duration delay> 默认 2 秒 <ch> 必须指定 A/D 通道号 <file path> 默认是 output 目录
ciTotAnalysis	ChronosInst in：ChronosInst 实例 duration second：数据分析持续时间 ch：指定的通道号	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	TOT 数据分析并保存数据到文件 ● Process 分析数据文件	<file path> 默认是 output 目录

	file path：数据文件目录 file name：数据文件名 error in：错误输入			
ciFetchTotProcessData	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 TOT Process 数据	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：tot, counts
ciStopTotAnalysis	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	停止 TOT 数据分析	
totAnalysisToFile	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 duration delay：分析延迟时间 ch：指定的通道号 file path：数据文件目录 file name：数据文件名 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	TOT 数据分析并保存数据到文件 ● Process 分析数据文件	<duration delay> 默认 2 秒 <file path> 默认是 output 目录
ciESAnalysis	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 ch：指定的通道号 A/D integration point: A/D 面积积分值	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	ES 数据分析并保存数据到文件 ● Process 分析数据文件	<ch> 必须指定 A/D 通道号 <file path> 默认是 output 目录

	A/D area scale: A/D面积积分缩放类型 file path：数据文件目录 file name：数据文件名 error in：错误输入			
ciFetchESProcessData	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 ES Process 数据	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：adc, counts
ciStopESAnalysis	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	停止 ES 数据分析	
esAnalysisToFile	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 duration delay：分析延迟时间 ch：指定的通道号 A/D integration point: A/D面积积分值 A/D area scale: A/D面积积分缩放类型 file path：数据文件目录 file name：数据文件名	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	ES 数据分析并保存数据到文件 ● Process 分析数据文件	<duration delay> 默认 2 秒 <ch> 必须指定 A/D 通道号 <file path> 默认是 output 目录

	error in：错误输入			
ciConformESAnalysis	ChronosInst in：ChronosInst 实例 duration second：数据分析持续时间 ch1：通道 1 的通道号 ch2：通道 2 的通道号 A/D integration point: A/D 面积积分值 A/D area scale: A/D 面积积分缩放类型 energy window min1：能量窗口最小值 1 energy window max1：能量窗口最大值 1 energy window min2：能量窗口最小值 2 energy window max2：能量窗口最大值 2 coins time window：符合时间窗值 avg_times:开启平均触发模式后的平均次数 file path：数据文件目录 file name：数据文件名 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	Conform ES 数据分析并保存数据到文件 ● Process 分析数据文件 ● ch1 Area Process 分析数据文件 ● ch2 Area Process 分析数据文件	<ch1><ch2> 必须指定不同的 A/D 通道号 <file path> 默认是 output 目录 <energy window min1> <energy window max1> <energy window min2> <energy window max2> 能量窗的有效值范围：0-16384 <avg times> 大于 0，启用平均触发模式；小于等于 0，默认是单次触发模式
ciFetchConformESP	ChronosInst in:	ChronosInst out:	实时获取	1）Process 数

rocessData	ChronosInst 实例 error in：错误输入	ChronosInst 实例 data： Int64 数值数组 error out：错误输出	Conform ES Process 数据	据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：tdiff, counts
ciFetchConformESArea1ProcessData	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 Conform ES Area1 Process 数据（低通道号）	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：adc, counts
ciFetchConformESArea2ProcessData	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 Conform ES Area2 Process 数据（高通道号）	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：adc, counts
ciStopConformESAnalysis	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	停止 Conform ES 数据分析	
conformESAnalysisToFile	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 duration delay：分析延迟时间 ch1：通道 1 的通道号 ch2：通道 2 的通道	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	ES 数据分析并保存数据到文件 ● Process 分析数据文件 ● ch1 Area Process 分析数据文件	<duration delay> 默认 2 秒 <ch1><ch2> 必须指定不同的 A/D 通道号 <file path> 默认是 output

	号 A/D integration point: A/D 面积积分值 A/D area scale: A/D 面积积分缩放类型 energy window min1：能量窗口最小值 1 energy window max1：能量窗口最大值 1 energy window min2：能量窗口最小值 2 energy window max2：能量窗口最大值 2 coins time window：符合时间窗值 avg_times:开启平均触发模式后的平均次数 file path：数据文件目录 file name：数据文件名 error in：错误输入		● ch2 Area Process 分析数据文件	目录 <avg times> 大于 0，启用平均触发模式；小于等于 0，默认是单次触发模式
ciPeriodAnalysis	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 ch：指定的通道号 sampling interval(ms)：采样间隔时间（单位：毫秒）	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	Period 数据分析并保存数据到文件 ● RAW 实时数据文件 ● Process 分析数据文件	<file path> 默认是 output 目录

	sampling number: 采样数量 file path：数据文件目录 file name：数据文件名 error in：错误输入			
ciFetchPeriodRAWData	ChronosInst in: ChronosInst 实例 timeout(ms)：超时时间（单位：毫秒） error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 Period RAW 数据	1）在超时时间内，直到获取数据才返回； 2）返回扁平 Int64 数组，二个数字一组，格式为：time, amplitude
ciFetchPeriodProcessData	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 Period Process 数据	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：period, counts
ciStopPeriodAnalysis	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	停止 Period 数据分析	
periodAnalysisToFile	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 duration delay：分析延迟时间 ch：指定的通道号 sampling interval(ms)：采样	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	Period 数据分析并保存数据到文件 ● RAW 实时数据文件 ● Process 分析数据文件	<duration delay> 默认 2 秒 <file path> 默认是 output 目录

	间隔时间（单位：毫秒） sampling number: 采样数量 file path：数据文件目录 file name：数据文件名 error in：错误输入			
ciStartStopAnalysis	ChronosInst in: ChronosInst 实例 duration second: 数据分析持续时间 start-stop chan groups：开始终止通道分组队列 dual stops chan groups: 双终止通道分组队列 coins time window：符合时间窗值 file path：数据文件目录 file name：数据文件名 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	Start-stop 数据分析并保存数据到文件 ● Dual Stops RAW 实时数据文件 ● Start-stop Process 分析数据文件 ● Start-stop Average Process 分析数据文件	<file path> 默认是 output 目录
ciFetchStartStopDualStopsRAWData	ChronosInst in: ChronosInst 实例 start chan:开始通道号 timeout(ms)：超时时间（单位：毫秒） error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取双终止通道 RAW 数据	1）在超时时间内，直到获取数据才返回； 2）返回扁平 Int64 数组，二个数字一组，格式为：x_stop_tdiff, y_stop_tdiff
ciFetchStartStopPro	ChronosInst in:	ChronosInst out:	实时获取开始终	1）Process 数

cessData	ChronosInst 实例 start chan：开始通道号 stop chan：终止通道号 error in：错误输入	ChronosInst 实例 data： Int64 数值数组 error out：错误输出	止通道 Process 数据	据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：time, counts
ciFetchStartStopAverageProcessData	ChronosInst in：ChronosInst 实例 start chan：开始通道号 error in：错误输入	ChronosInst out：ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取开始终止通道 Average Process 数据	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：time, counts
ciStopStartStopAnalysis	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	停止 Start-stop 数据分析	
startStopAnalysisToFile	ChronosInst in：ChronosInst 实例 duration second：数据分析持续时间 duration delay：分析延迟时间 start-stop chan groups：开始终止通道分组队列 dual stops chan groups： 双终止通道分组队列 coins time window：符合时间窗值 file path：数据文件目录	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	Start-stop 数据分析并保存数据到文件 ● Dual Stops RAW 实时数据文件 ● Start-stop Process 分析数据文件 ● Start-stop Average Process 分析数据文件	<duration delay> 默认 2 秒 <file path> 默认是 output 目录

	file name：数据文件名 error in：错误输入			
ciClearAnalysisData	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	清除分析缓存数据	

9.5.12 注意事项

1) 状态码：

定义	值	说明
CI_SUCCESS	0	操作成功
CI_FAIL	-1	操作失败
CI_TRUE	1	布尔值 True
CI_FALSE	0	布尔值 False

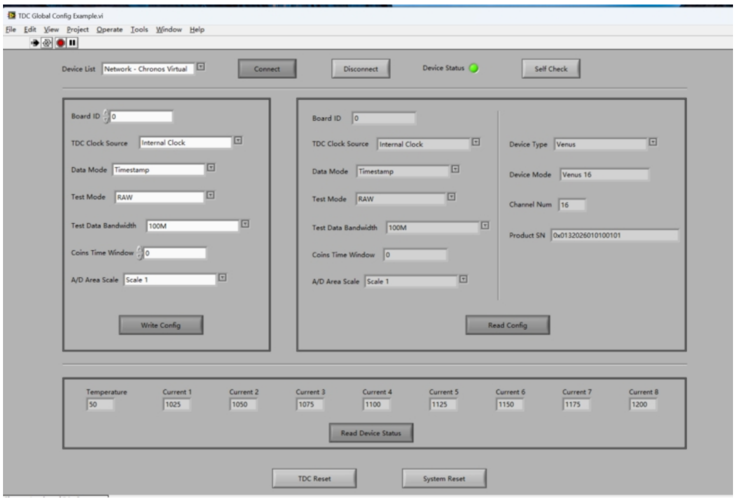
2) 线程安全：

- 所有 API 调用都是线程安全的；
- 数据采集和数据分析操作是异步的；
- 每次数据采集和分析完，建议手动调用 stop API。

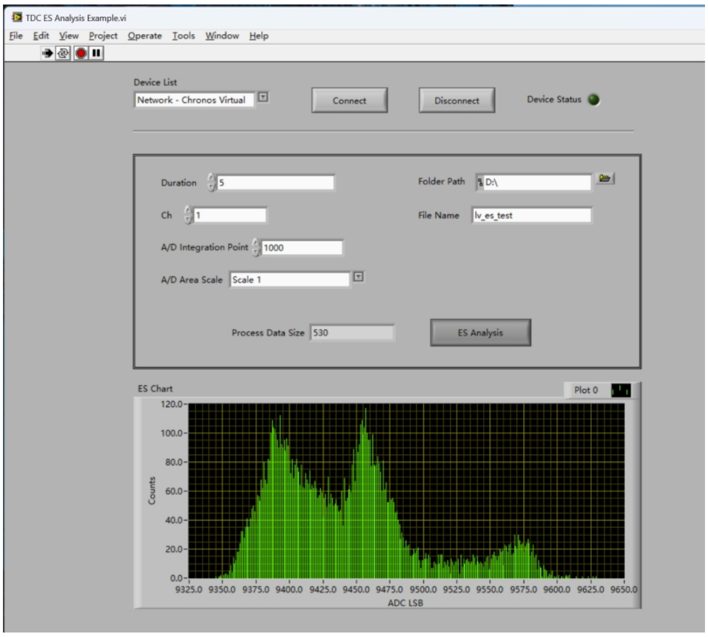
9.5.13 参考示例

请参考 Examples 目录下的示例。

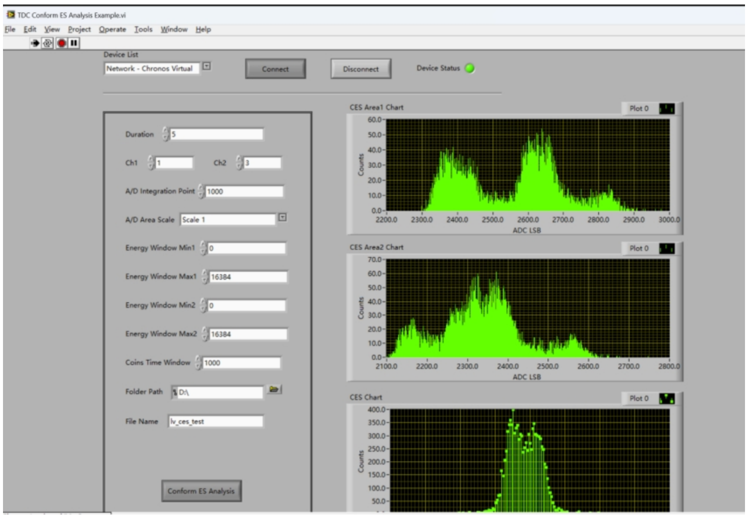
1) 全局配置 Example



2) 能谱分析 Example



3) 符合能谱分析 Example



9.6 原始数据离线分析参考（Matlab）

9.6.1 MATLAB 原始数据分析参考脚本

官方提供针对裸数据的分析 MATLAB 脚本函数，用户可以调用或者参考进行二次开发。文件名称为“Venus_Raw_data_analysis_v1.0.0”。用户运行此函数（MATLAB 版本需要 MATLAB 2020a 以上），导入采集到的原始数据，然后程序会自动分析并生成转换后的结果。

比如，分析时间全局符合数据的步骤：

（1） 数据导入；

（2） 函数运行；

（3） 输出结果；

表格 1 MATLAB 原始数据分析脚本输出结果说明

原始数据类型	分析后输出文件名称	文件格式
时间符合数据	global_coins.txt	通道号 1-通道号 2-时间差
脉宽测量数据	tot.txt	通道号-脉宽值
ADC 测量数据	adc.txt	通道号-AD 值
能谱测量数据	area.txt	通道号-时间戳-能量
时间-能谱符合数据	Area_coins.txt	通道号 1-通道号 2-能量 1-能量 2-时间差
双边沿时间戳数据	Dual_singles.txt	通道号-边沿类型-时间戳

时间戳数据	Time_tagger.txt	通道号-时间戳
周期数据	Period.txt	周期
带时间戳的全局符合数据	Glbt_coins.txt	通道号 1-通道号 2-通道号 1 的时间戳

9.6.2 MATLAB 分析脚本组成

官方提供的 MATLAB 分析程序针对用户采集到的原始测量裸数据进行分析，用户可以使用此分析程序，也可以参考进行二次开发。

分析程序包括四个文件夹，分别是：

（1）./dat： 保存用户使用上位机软件采集到的 Raw data；

（2）./function：保存分析程序，包括：

（2.1）dat_proc.m：主程序；

（2.2）dat_type_proc.m：提取 Raw data 并分析数据中是否存在错误信息；

（2.3）dat_Raw_proc.m：将 Raw data 进行分类，提取各种数据类型的数据，包括时间戳、符合时间、ADC、能谱等；

（2.4）write_results.m：将分析结果保存在./results 文件夹中。

（3）./results：包括./figure 和./txt 两个文件夹，分别保存产生的分析结果图表和数据；

（4）./docx：包括 MATLAB 分析程序使用说明和其他相关文档。

9.6.3 分析程序使用方法

建议用户采用如下步骤使用分析程序：

（1） 首先，打开 MATLAB 2020a 以上版本软件；

（2） 打开主程序 dat_proc.m，点击“run”；

（3） 选择要分析的 Raw data 文件（dat 格式或者 bin 格式）；

（4） 运行完成后，在./results 文件夹中得到分析结果。

9.6.4 运行举例

（1）输入函数主路径，打开“data_proc.m”脚本，点击“运行”；

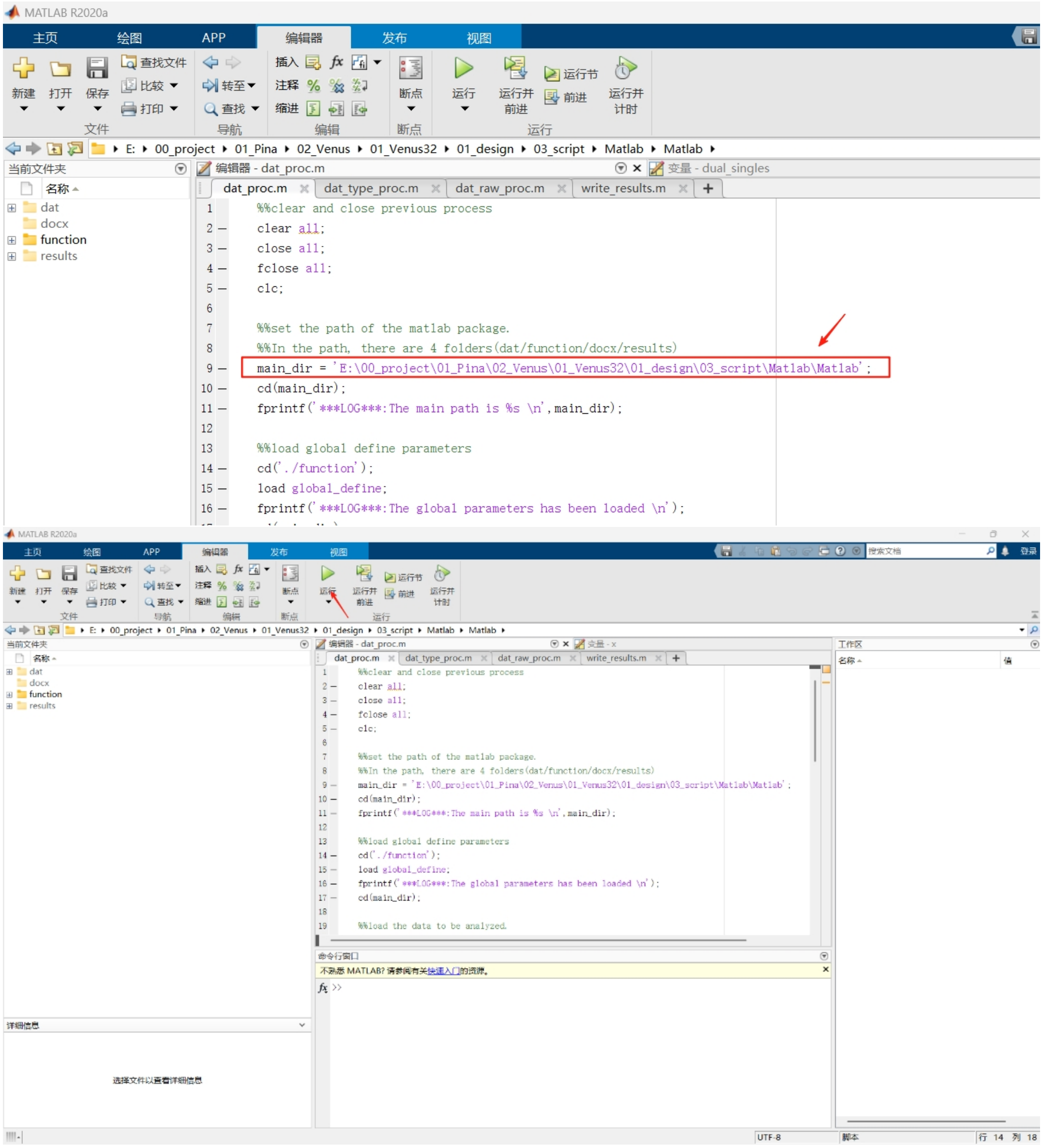


图 1 数据分析 MATLAB 脚本运行

（2）弹出文件选择对话框，选择要分析的 Raw 数据文件，点击“打开”；

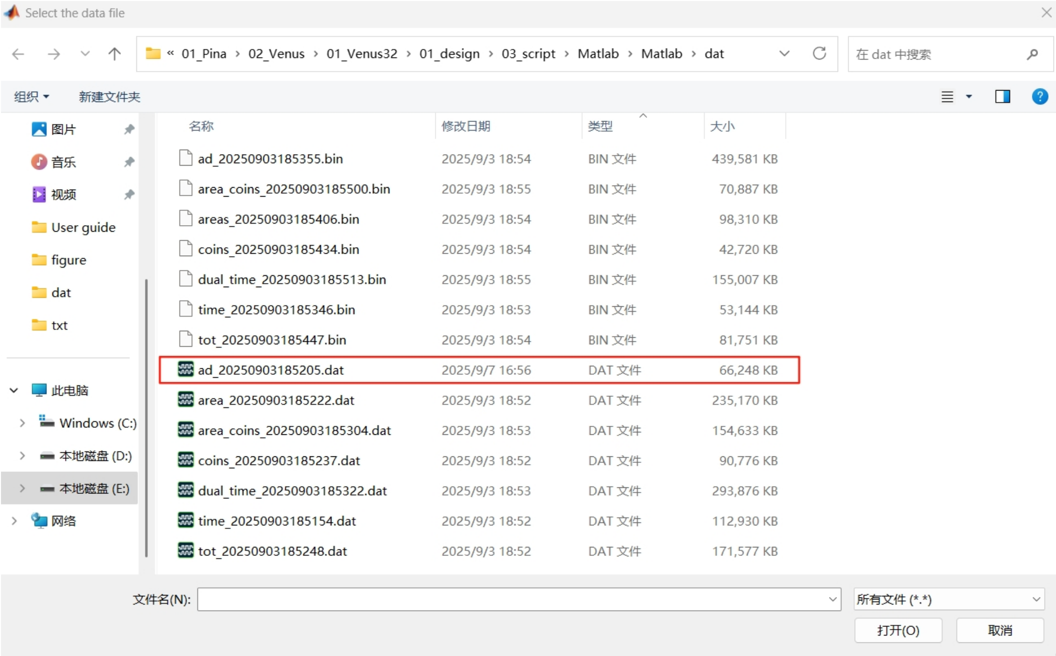


图 2 选择要分析的 adc Raw 数据文件

（3）程序会自动完成数据读取，格式转换和结果保存等；

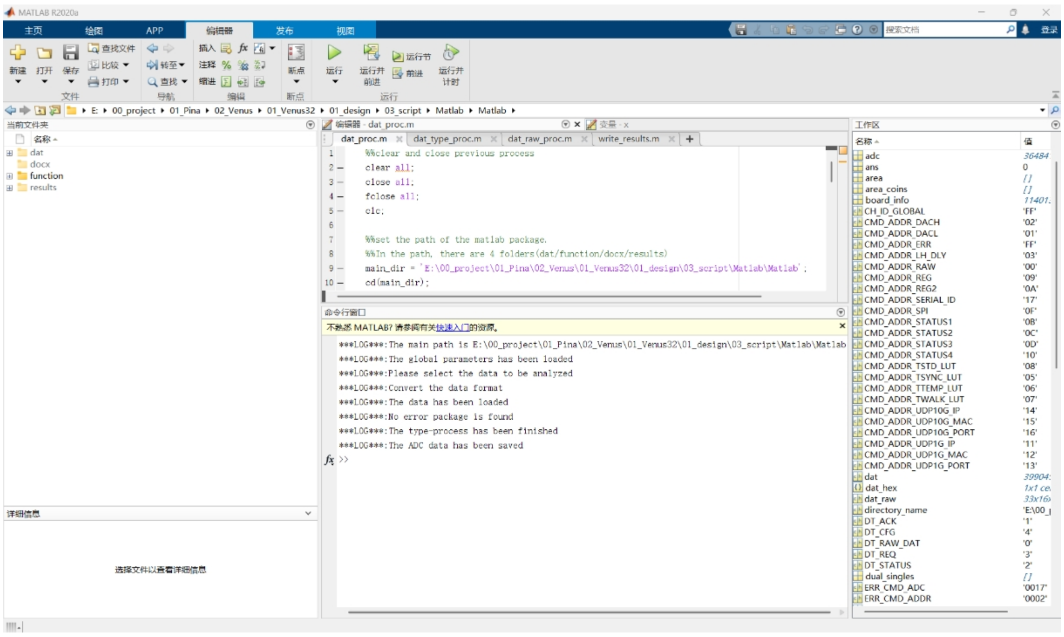


图 3 数据分析 MATLAB 脚本运行

（4）程序会自动完成数据读取，格式转换和结果保存等；

对于 A/D 波形数据，第一列为通道号，其他列为 ADC 计数值，每行按照时间排列，采样周期为 20 ns。

adc. txt

```
Channel-A ADC Channel-B ADC
13 2212 11 2207
1 2196 15 2215
5 2207 3 2214
9 2200 7 2208
13 2210 11 2200
1 2194 15 2211
5 2199 3 2209
9 2194 7 2202
13 2203 11 2194
1 2187 15 2204
5 2194 3 2202
9 2192 7 2196
13 2199 11 2190
```

图 4 输出“adc.txt”数据格式

对于过阈时间数据，第一列为通道号，其他列为 TOT 计数值，单位为 ns。

tot. txt

```
Channel TOT(ns)
14.000000 134.781075
8.000000 131.768325
9.000000 131.053650
10.000000 135.608850
15.000000 137.929350
1.000000 131.168700
2.000000 135.746325
3.000000 131.014650
0.000000 137.344350
5.000000 131.918475
11.000000 131.628900
12.000000 131.149200
13.000000 131.302275
4.000000 135.858450
```

图 5 输出“tot.txt”数据格式

对于面积数据，第一列为通道号，第二列为时间戳计数值，单位为 ns，第三列为 area 值，单位为 ADC LSB。

area.txt

```
Channel Time tagger(ns) Area(ADC)
11.000000 97730237132.774994 6755.000000
5.000000 97730235812.625000 6913.000000
7.000000 97730237151.300003 7244.000000
9.000000 97730242563.524994 7078.000000
1.000000 97730241241.425003 8309.000000
3.000000 97730240392.199997 8956.000000
13.000000 97730236531.199997 9688.000000
15.000000 97730236695.000000 10120.000000
11.000000 97731235517.175003 6742.000000
5.000000 97731234212.625000 6880.000000
7.000000 97731235551.300003 7238.000000
9.000000 97731240963.524994 7068.000000
1.000000 97731239641.425003 8328.000000
3.000000 97731238784.399994 8960.000000
13.000000 97731234931.199997 9681.000000
15.000000 97731235095.000000 10132.000000
```

图 6 输出 “area.txt” 数据格式

对于参考符合或者全局符合数据，第一列为通道号 1，第二列为通道号 2，第三列为通道号 2 与通道号 1 的时间差值（T2-T1），单位为 ns。

global_coins.txt

```
Channel-A Channel-B Delt_T(ns)
12.000000 14.000000 15.600000
6.000000 14.000000 54.600000
5.000000 14.000000 78.975000
4.000000 14.000000 482.625000
13.000000 14.000000 813.150000
15.000000 14.000000 967.200000
11.000000 14.000000 1404.975000
7.000000 14.000000 1415.700000
3.000000 14.000000 4673.175000
2.000000 14.000000 5157.750000
1.000000 14.000000 5533.125000
0.000000 14.000000 6162.000000
10.000000 14.000000 6721.650000
```

图 7 输出 “global_coins.txt” 数据格式

对于能谱符合数据，第一列为通道号 1，第二列为通道号 2，第三列为通道 1 的信号面积，第四列为通道 2 的信号面积，第五列为通道 2 与通道号 1 的时间差

值（T2-T1），单位为 ns。

area_coins.txt

```
Channel-A Channel-B Area-A(ADC) Area-B(ADC) Delt_T(ns)
11.000000 15.000000 6754.000000 10135.000000 431.925000
7.000000 15.000000 7176.000000 10135.000000 448.500000
3.000000 15.000000 8947.000000 10135.000000 3683.550000
1.000000 15.000000 8327.000000 10135.000000 4544.475000
9.000000 15.000000 7033.000000 10135.000000 5839.275000
7.000000 11.000000 7176.000000 6754.000000 16.575000
3.000000 11.000000 8947.000000 6754.000000 3251.625000
1.000000 11.000000 8327.000000 6754.000000 4112.550000
9.000000 11.000000 7033.000000 6754.000000 5407.350000
3.000000 7.000000 8947.000000 7176.000000 3235.050000
1.000000 7.000000 8327.000000 7176.000000 4095.975000
9.000000 7.000000 7033.000000 7176.000000 5390.775000
1.000000 3.000000 8327.000000 8947.000000 860.925000
9.000000 3.000000 7033.000000 8947.000000 2155.725000
9.000000 1.000000 7033.000000 8327.000000 1294.800000
13.000000 5.000000 9680.000000 6796.000000 741.975000
```

图 8 输出“area_coins.txt”数据格式

对于双沿时间戳数据，第一列为通道号，第二列为边沿类型（0：上升沿，1：下降沿），第三列为时间戳计数值，单位为 ns。

dual_singles.txt

```
Channel Edge Time tagger(ns)
4.000000 0.000000 96383350259937.000000
4.000000 1.000000 96383349397634.328125
5.000000 0.000000 96383350259535.296875
5.000000 1.000000 96383349393321.890625
6.000000 0.000000 96383350259500.203125
6.000000 1.000000 96383349398301.218750
7.000000 0.000000 96383350260889.578125
7.000000 1.000000 96383349396905.015625
11.000000 0.000000 96383350260873.968750
8.000000 1.000000 96383349399943.125000
12.000000 0.000000 96383350259483.625000
```

图 9 输出“dual_singles.txt”数据格式

对于周期测量数据数值，单位为 ns。

period.txt

```
Period(ns)
998.396466
998.396466
998.394516
998.394516
998.395003
998.395003
998.394881
998.394881
998.395612
998.395612
998.395247
998.395247
998.394759
```

图 10 输出“period.txt”数据格式

对于带时间戳的全局符合测量数据，第一列为通道号 1，第二列为通道号 2，第三列为通道 1 的时间戳，单位为 ns。

glbt_coins.txt

```
Channel-A Channel-B Timestamp of Channel-A(ns)
12.000000 10.000000 816537738594240.000000
2.000000 10.000000 816537738594240.000000
1.000000 10.000000 816537738594240.000000
15.000000 10.000000 816537738594240.000000
14.000000 10.000000 816537738594240.000000
3.000000 10.000000 816537738594240.000000
16.000000 10.000000 816537738594240.000000
5.000000 10.000000 816537738594240.000000
6.000000 10.000000 816537738594240.000000
0.000000 10.000000 816537738594240.000000
9.000000 10.000000 816537738594240.000000
4.000000 10.000000 816537738594240.000000
11.000000 13.000000 816537738594240.000000
7.000000 13.000000 816537738594240.000000
12.000000 13.000000 816537738594240.000000
2.000000 13.000000 816537738594240.000000
1.000000 13.000000 816537738594240.000000
```

图 11 输出”glbt_coins.txt”数据格式

9.7 原始数据离线分析参考（Python）

与 Matlab 分析参考脚本类似，Python 脚本组成如下表所示。

表格 2 Raw 数据分析参考脚本说明

文件夹	内容	函数列表	说明
./dat	待分析的 Raw 数据		
./docx			
./function	.py 脚本程序	run_all.py	运行脚本
		matlab_converter.py	主函数
		global_define.py	全局参数脚本
		dat_type_proc.py	数据类型处理脚本
		dat_raw_proc.py	Raw 数据处理脚本
		write_results.py	结果生成和写入脚本
./results	输出结果		

打开 Powershell，运行 python run_all.py 函数；

然后，程序会让用户选择待分析的数据；

然后，程序会自动化运行并输出结果，如下图所示。输出的结果与 Matlab 脚本一致。

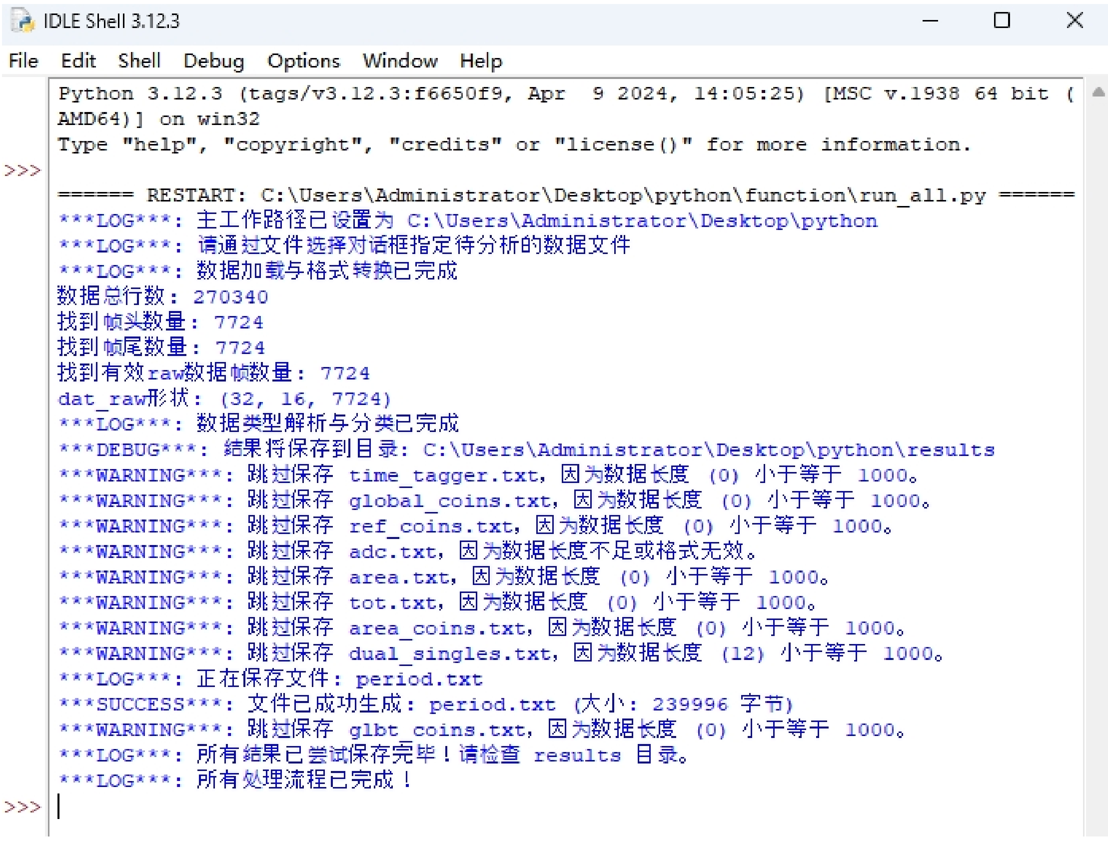


图 12 Raw 数据分析 python 脚本分析过程

本文档历史记录

文档编号	修改时间	修改人	说明
UG-01-3-010-1	2025.12	Cong	V1.0
UG-01-3-030-1	2026.2	Cong	1. 针对 v1.2 硬件版本进行更新； 2. 优化了时间测量性能； 3. 增加了比较器迟滞电压设置功能； 4. 去掉了时间延迟使能按钮； 5. 修改了外部时钟输入要求； 6. 增加了 LabVIEW SDK，更新了 C++/Python/Matlab SDK； 7. 增加了 Start-Stop 分析模式； 8. 更新了部分测试结果
UG-01-3-040-1	2026.4	Cong	1. 针对 v1.3 硬件版本进行更新； 2. 详细描述了 Start-stop 统计方法； 3. 更新了产品订购型号说明； 4. 更新了测试结果和 API 接口
UG-01-3-050-1	2026.6	Cong	1. 增加了关联函数 g2 计算； 2. 优化了时钟频率分析功能，满足 IEEE 1139； 3. 增加了多重符合统计； 4. 解决了部分 bug；