Chronos

Venus lite series.
32/24/16/8 Channel, 0.975 ps
Streaming Time-Digital Converter

Venus Lite User Guide

UG-01-3-030-1, Feb.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*

Chronos

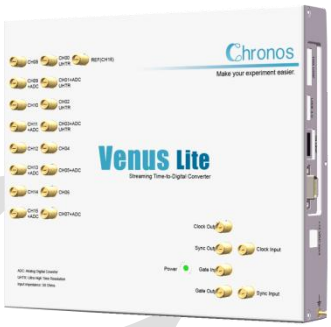
目录

图目录

表目录

1. 产品特征

功能丰富



- 32/24/16/8 通道时间-数字转换器（TDC）
- 16/8 通道模拟-数字转换器（ADC）
- 32/24/16/8 通道过阈脉宽测量（TOT）
- 边沿定时电路，高精度阈值电压设置
- 多种测量模式：时间戳/在线全局符合/能谱/波形测量
- 内/外部时钟和同步输入，支持多达 256 个设备组网（超过 4000 个测量通道）
- 内置 4G DDR 数据缓存
- USB3.0/1G 以太网/10 G 以太网数据接口
- 提供 40 Gbps QSFP 等定制接口协议
- 配套数据采集上位机软件，配套数据处理软件包
- 开放 C/C++/MATLAB/Python/Labview API 接口
- 支持在线计数强度/时间差直方图/衰变时间/能谱分析/频率分析功能
- 支持自动模拟前端噪声、基线标定
- 220V 交流/12 V 直流输入供电

高性能

- 0.975 ps 时间测量分辨率
- 单通道时间测量精度~ 3 ps RMS
- 通道时间延迟可配置
- 单通道时间测量死时间~ 2 ns，可配置
- 50 Msps, 12 bit ADC 采样
- 模拟前端输入范围-2 V - 2 V, 50 Ohms 输入阻抗
- 可设置甄别阈值范围-2 V - 2 V，14 bit

其他

- 多链 FPGA-TDC 技术
- 实时温度/电流监控
- 支持固件/软件远程更新，支持特殊功能定制服务

2. 技术规格

+12 V 直流供电，本地时钟，测试信号来自信号发生器，信号幅度 2.5 V，上升沿 1 ns，重复频率 1 MHz，经过功分器输入到 Venus 设备两个通道中。对双通道时间差进行高斯拟合，得到其均方根值为σ，则单通道时间分辨率为σ/√2.

表格 1 Venus 系列系统技术规格参数

参数	Venus 32/24	Venus lite 16/8	Venus lite-T 16/8	单位	测试条件/说明
建议工作环境温度	0/40	0/40		℃ min/max	
系统散热	风冷	被动散热	风冷		
风扇个数	4	/	2		
风扇转速	4300	/	4300	RPM	风扇自动调速
气流	9.7 x 4	/	9.7 x 2	CFM	
噪声	22 x 4	/	22 x 2	dB	
测量通道数					
时间测量通道数	32 或 24	16 或 8	16 或 8	个	参考产品列表 A
AD 测量通道数	16	8	8		
TOT 测量通道数	32 或 24	16 或 8	16 或 8		
参考测量通道					
通道数	1	1	1	个	参考测量通道输入数字脉冲信号，不经过模拟前端，其他功能与正常测量通道无异
输入阻抗	50±1%	50±1%	50±1%	Ω	
输入信号高电平	2.4/3.3	2.4/3.3	2.4/3.3	Vmin/max	
输入信号低电平	2.4/3.3	2.4/3.3	2.4/3.3	Vmin/max	
外部时钟输入					
标准输入频率	25	25	25	MHz	
输入信号高电平	2.4/3.3	2.4/3.3	2.4/3.3	Vmin/max	
输入信号低电平	0/0.8	0/0.8	0/0.8	Vmin/max	
外部同步输入					
最小高电平宽度	40	40	40	ns	高电平有效 B
输入信号高电平	2.4/3.3	2.4/3.3	2.4/3.3	Vmin/max	
输入信号低电平	0/0.8	0/0.8	0/0.8	Vmin/max	
系统输出时钟					
同步输出通道数	1	1	1	个	
输出时钟频率	25	25	25	MHz	
模拟前端 C					
增益	+1	+1	+1	V/V	
输入阻抗	50±1%	50±1%	50±1%	Ω	
输入电容	5	5	5	pF	

输入信号范围					
正向脉冲	0/+2	0/+2	0/+2	Vmin/max	仅做时间测量时，可以输入更大幅度信号，建议范围不超过±5 V
负向脉冲	-2/0	-2/0	-2/0	Vmin/max	
可设阈值					
范围	-2/2	-2/2	-2/2	Vmin/max	
分辨率	0.6	0.6	0.6	mV	
TDC 参数					
测量最小刻度	0.975	0.975	0.975	ps	
在线非线性修正	是	是	是		LSB is 0.975 ps
DNL(高分通道)	/	-1 ~ 5	-1 ~ 5	LSB	
DNL(普通通道)	-1 ~ 15	-1 ~ 15	-1 ~ 15	LSB	
INL(高分通道)	/	<±5	<±5	LSB	
INL(普通通道)	<±20	<±20	<±20	LSB	
ADC 参数					
采样率	50	50	50	Msps	
分辨率	12	12	12	Bits	
INL	<±1	<±1	<±1	LSB	
DNL	<±0.65	<±0.65	<±0.65	LSB	Fin=10 MHz
SNR	70	70	70	dB	
ENOB	11.3	11.3	11.3	bits	
SINAD	70	70	70	dB	
单通道时间测量 D					
普通通道抖动	~6.0	~6.0	~6.0	ps RMS, σ/√2	
高分辨率通道抖动	/	~3.0	~3.0	ps RMS, σ/√2	高分辨率通道指Ch0-3 内部触发测试
单通道脉宽测量 D					
最小测量脉宽	1	1	1	ns	高分辨率通道指Ch0-3 内部触发测试
最大测量脉宽	16	16	16	us	
时间测量死时间					
死时间可设范围	2/100000	2/100000	2/100000	ns min/max	上位机软件可配置
死时间设置分辨	2	2	2	ns	
迟滞电压					
可设置档位	1/30/40/70	1/30/40/70	1/30/40/70	mV	上位机软件可配置
通道时间延迟					
通道延迟可设范围	0/1000000	0/1000000	0/1000000	ns	

通道延迟设置分辨	0.975	0.975	0.975	ps	
输出数据类型分类 [E]					用户可以采集任一数据类型的裸数据
时间参考符合数据	支持	支持	支持		
时间全局符合数据	支持	支持	支持		
ADC 测量数据	支持	支持	支持		
面积测量数据	支持	支持	支持		
时间-能谱全局符合	支持	支持	支持		
双沿时间戳数据	支持	支持	支持		
过阈时间测量数据	支持	支持	支持		
周期测量数据	支持	支持	支持		
在线时间符合					
符合时间半窗范围	0/16000	0/16000	0/16000	ns min/max	
符合时间半窗分辨	0.975	0.975	0.975	ps	
在线能量积分					
积分最大时间	1.3	1.3	1.3	ms	
积分时间分辨	20	20	20	ns	
在线事件缓存	250	250	250	Mtags	1 tag 指 1 个时间戳或 ADC 波形数据事例
对外数据接口带宽 [F]					
USB 3.0	40	40	40	Mtags	
千兆以太网	25	25	25	Mtags	
万兆以太网	300	300	300	Mtags	
定制 4 路 QSFP	1200	1200	1200	Mtags	
系统供电 [G]					
直流电压/电流	12/>3	12/>3	12/>3	V/A	
直流供电纹波要求	<0.1	<0.1	<0.1	Vpp	
系统功耗	<40	<25	<25	W	
设备尺寸 [H]	314x314x100	190x170x42	190x190x73	WxLxH, mm^3	
外壳材料	镁铝合金	铝 6061	铝 6061		
丝印油墨材料	嘉宝莉 72	嘉宝莉 72	嘉宝莉 72		

说明：

A. 产品列表详见“产品订购选型”；
B. 外部时钟和同步说明详见“多设备时钟同步”；
C. 模拟前端电路结构详见“产品功能结构”；
D. 典型测试电路的连接方法及测试结果详见“典型测试电路和测试结果”；
E. Venus 支持各种数据类型裸数据采集，数据类型和结构详见“测量类型原始数据结构”；

F. 设备对外数据接口介绍详见“对外数据接口”；
G. 官方提供配套电源适配器，系统的供电接口详见“设备供电接口”；
H. 设备具体尺寸图详见“产品外观尺寸”。

3. 产品订购选型

表格 2 Venus 系列产品订购选型

产品系列	产品	时间测量通道数	AD 测量通道数	对外接口	供电方式
Venus[A]	Venus 32/24	32/24	16	USB3.0/千兆网/万兆网	220V 交流/12V 直流供电
	Venus lite 16/8	16/8[B]	8	USB3.0/千兆网/万兆网	12V 直流供电
	Venus lite-T 16/8 [C]	16/8[B]	8	USB3.0/千兆网/万兆网	12V 直流供电

说明：

A. 所有产品支持定制其他协议接口，并可定制模拟前端增益；
B. **Venus Lite** 产品的通道 CH0~CH3 为超高精度测量通道；
C. 对于工作环境为无对流环境时，用户应选择 Venus lite 产品，对于一般工作环境，建议选择 Venus lite-T 产品。

4. 产品功能结构

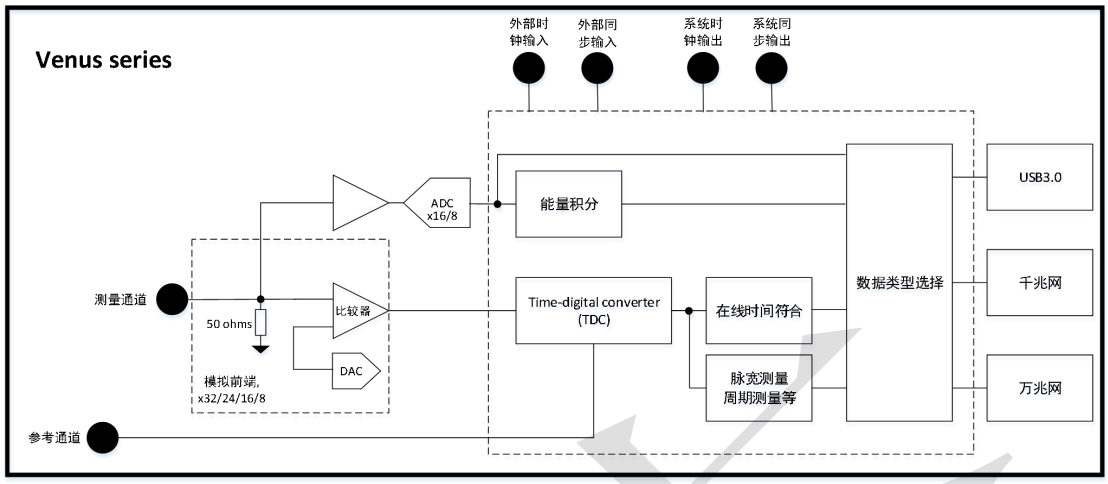


图 1 Venus 系列产品功能结构框图

5. 产品外观尺寸

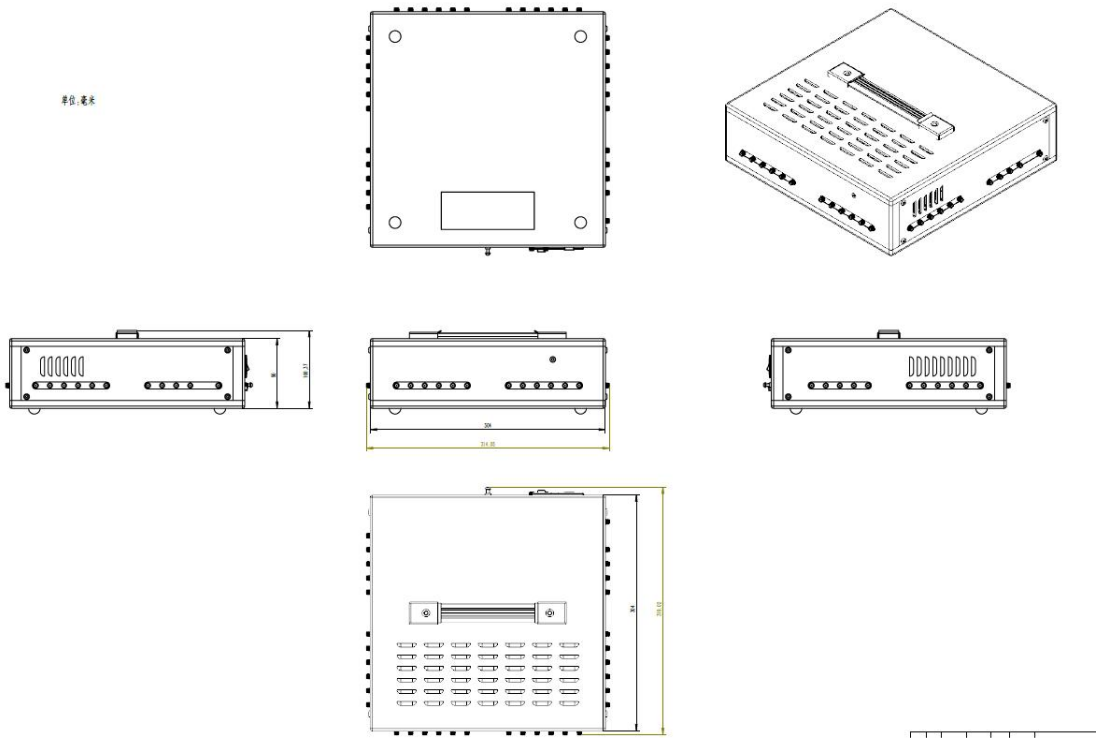


图 2 Venus 系列产品外观和结构尺寸图

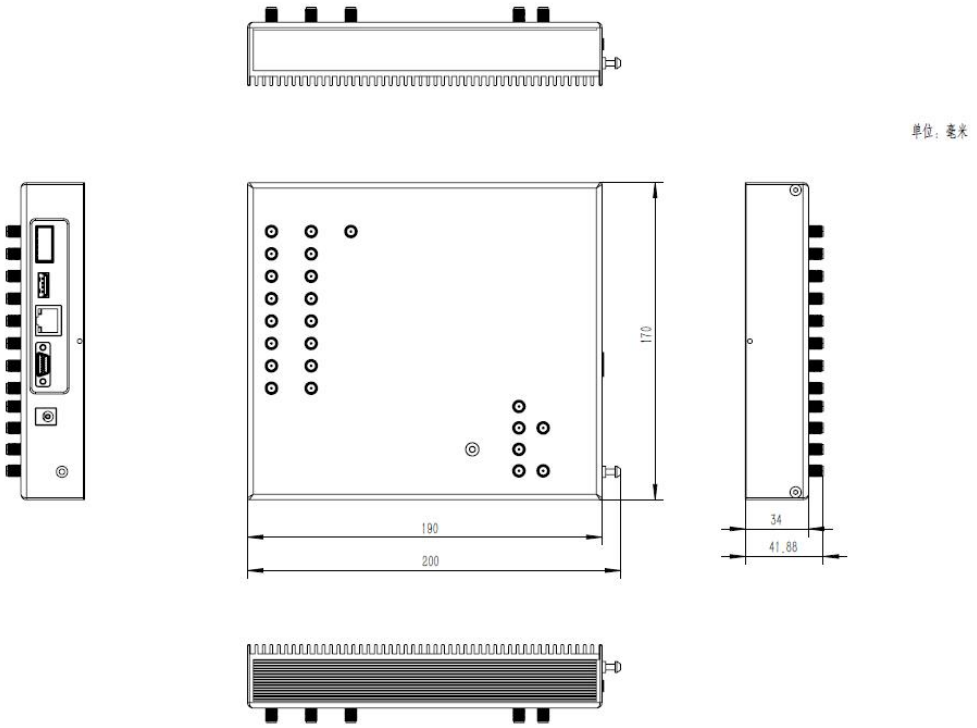


图 3 Venus Lite 系列产品外观和结构尺寸图

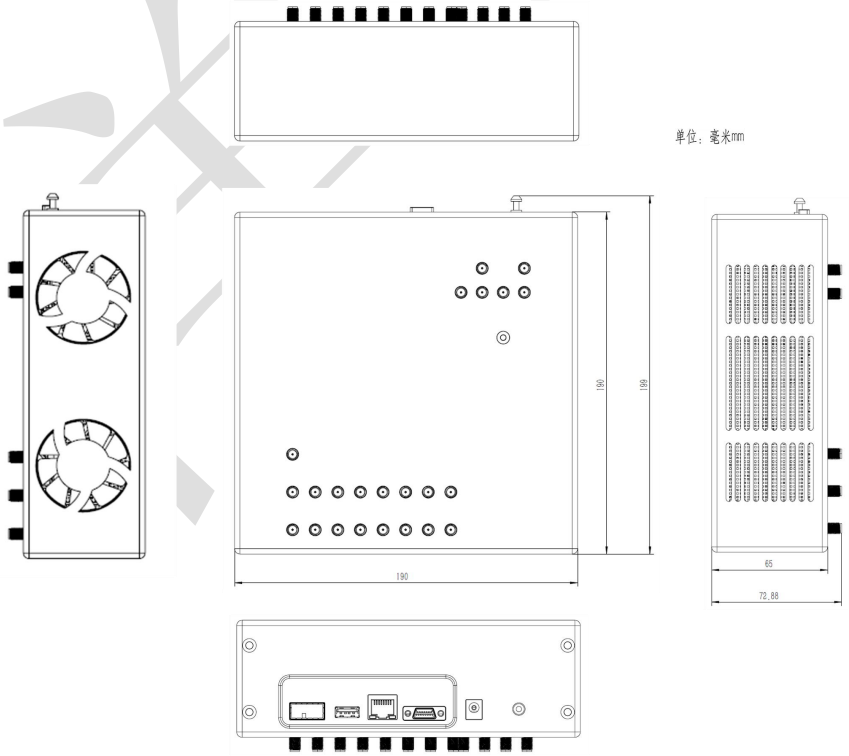


图 4 Venus Lite-T 系列产品外观和结构尺寸图

6. 绝对最大额定值

表格 3 Venus 设备工作额定值

参数	额定值	说明
电子		
交流供电	200 VAC ~ 260 VAC	
直流供电	+11 V ~ +15 V	
测量通道输入	-5 V ~ +5 V	
参考通道输入	0 V ~ +5 V	
外部时钟输入	0 V ~ +5 V	
外部同步输入	0 V ~ +5 V	
环境		
设备存储温度	-50 ℃ ~ 100 ℃	
设备工作温度	0 ℃ ~ 50 ℃	

7. 设备接地和静电

	测量通道对静电敏感，请勿用手直接接触测量通道 SMA 连接器的内芯。 请确保设备外壳与实验室大地良好接触（接地阻抗<1 Ω）。 请注意上位机电脑与待测探测器接地情况。 运输时，设备需使用防静电袋等材料妥善包装，以防静电击穿导致器件损坏。

8. 上位机软件使用说明

8.1 软件安装说明

官方提供名为 Venus TDC v1.0 的上位机软件安装包。请用户按照以下安装向导完成安装。

为避免安装过程中受到干扰，建议在安装前暂时**关闭系统的防火墙和杀毒软件**。

8.1.1 Windows 系统

8.1.1.1 软件安装步骤

（1）双击运行安装程序

显示“选择安装模式”界面时，可选择仅为当前用户或为所有用户安装。

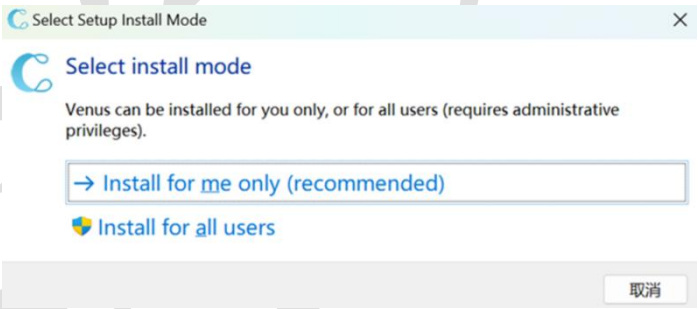


图 5 上位机安装步骤 1

（2）选择程序安装路径

选择安装路径。请注意：目标文件夹需具备管理员写入权限。选择后，点击“下一步”。

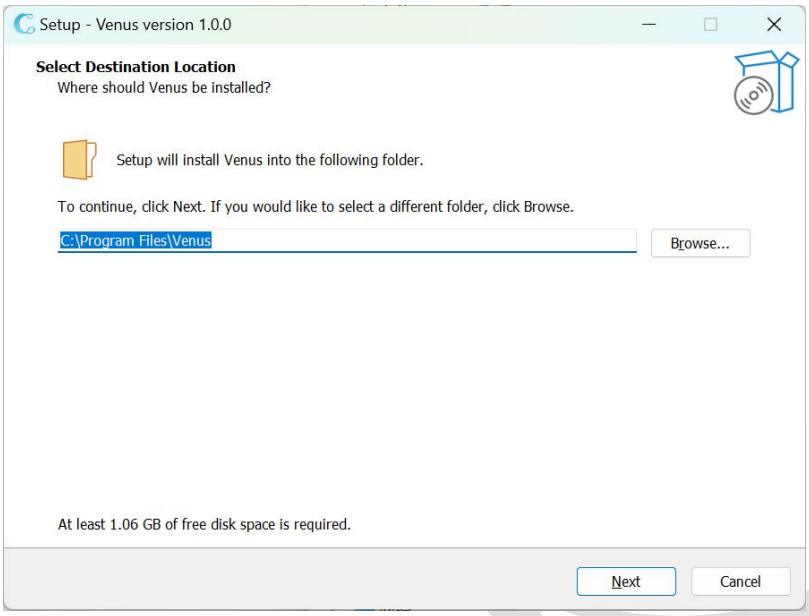


图 6 上位机安装步骤 2

（2）选择是否创建桌面快捷方式

点击“下一步”；

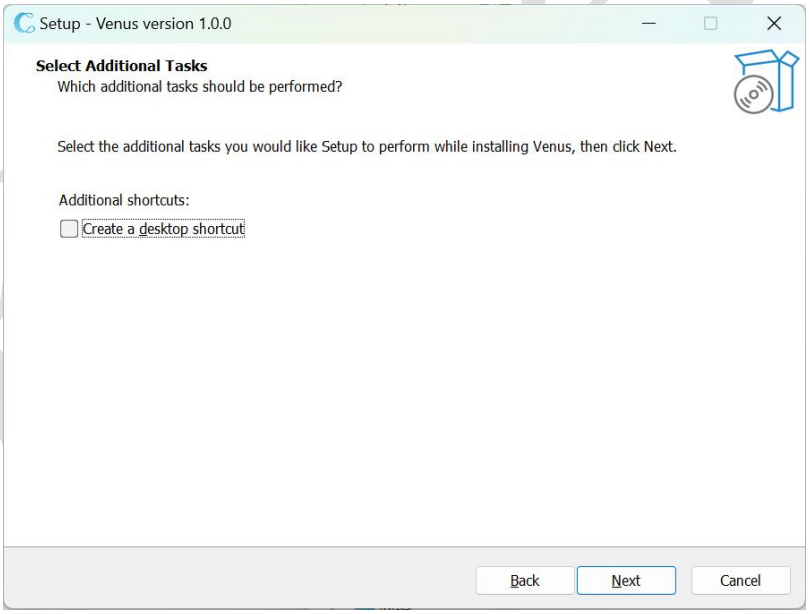


图 7 上位机安装步骤 3

（3）开始安装

点击“install”，将开始安装过程；

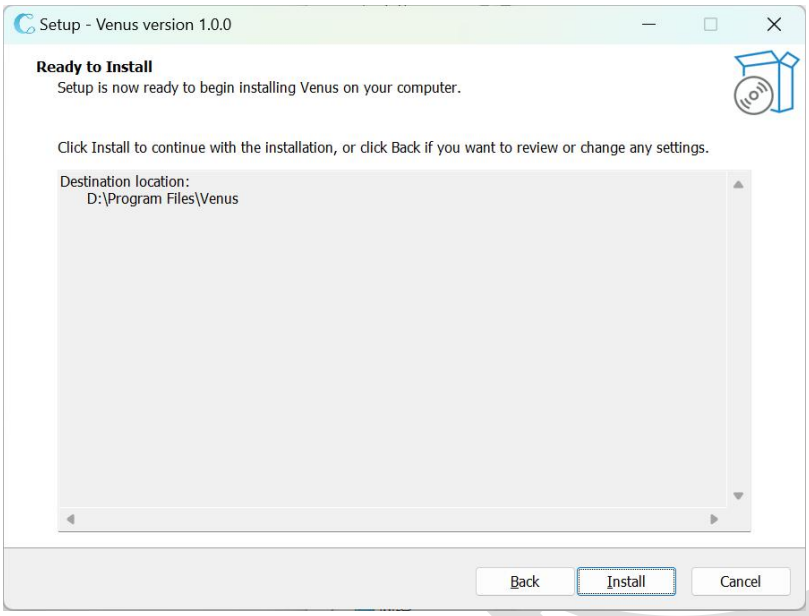


图 8 上位机安装步骤 4

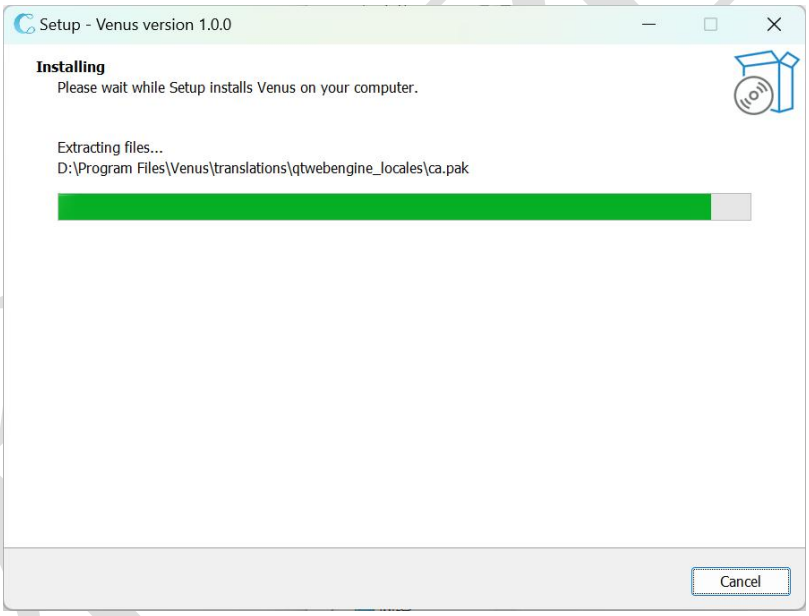


图 9 上位机安装步骤 4 安装过程

（4）安装成功

如果安装成功，将显示如下界面。

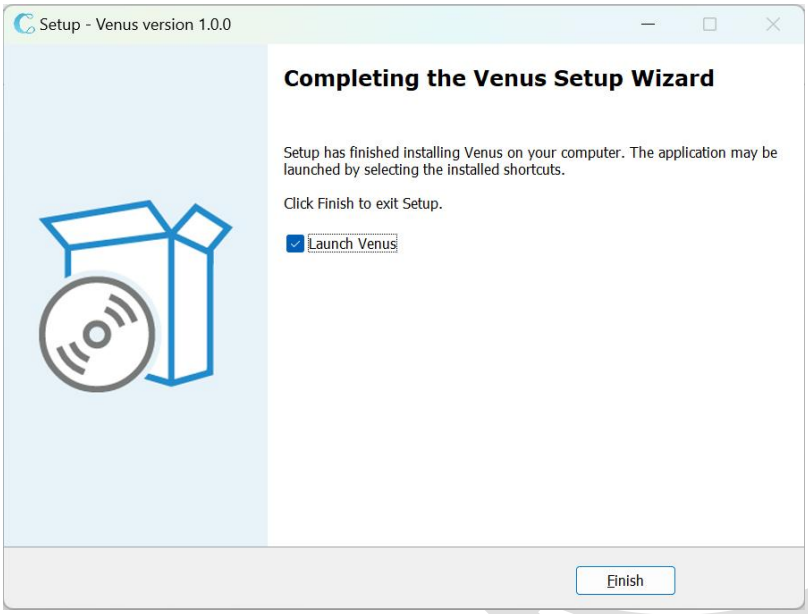


图 10 上位机安装完成

8.1.1.2 软件安装注意事项

（1）USB3.0 驱动安装

安装过程中，软件将静默安装 USB 3.0 驱动程序。设备上电后，若计算机无法识别设备，请首先确认 USB 3.0 驱动是否安装成功。如未成功，请进行手动安装。

（1.1）打开设备管理器，确认 USB 设备未识别

一般在其他设备列表中，会发现有黄色警告标识的未识别 USB 设备。

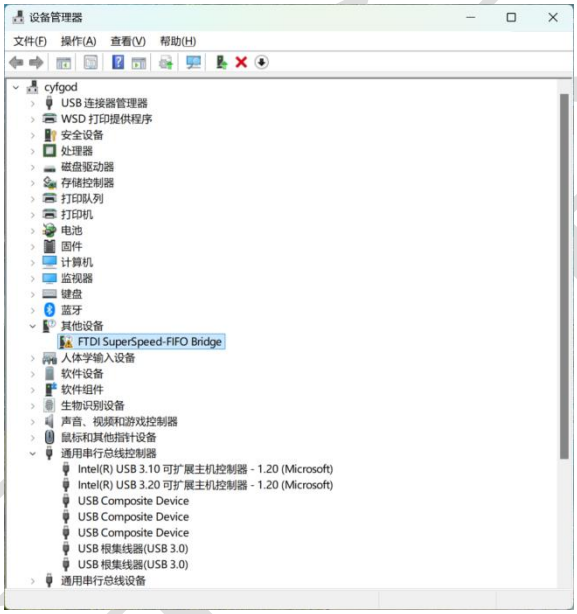
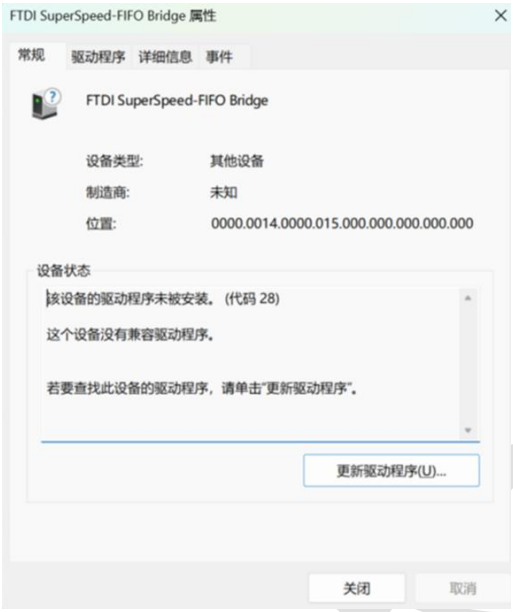


图 11 USB3.0 驱动设备管理器

（1.2）双击这个未识别设备，选择“更新驱动程序”

在搜索驱动程序对话框中，选择“浏览我的电脑以查找驱动程序”。导航至程序安装目录下的 \driver\FTD3XXDriver_WHQLCertified_v1.3.0.10\x64 文件夹。

按照后续提示完成安装。


FTDI SuperSpeed-FIFO Bridge 属性
常规
驱动程序
详细信息
事件
FTDI SuperSpeed-FIFO Bridge
设备类型:
其他设备
制造商:
未知
位置:
0000.0014.0000.015.000.000.000.000.000
设备状态
该设备的驱动程序未被安装。(代码 28)
这个设备没有兼容驱动程序。
若要查找此设备的驱动程序，请单击"更新驱动程序"。
更新驱动程序(U)...
关闭
取消


更新驱动程序 - FTDI SuperSpeed-FIFO Bridge
你要如何搜索驱动程序?
自动搜索驱动程序(S)
Windows 将在你的计算机中搜索最佳可用驱动程序，并将其安装在你的设备上。
浏览我的电脑以查找驱动程序(R)
手动查找并安装驱动程序。
取消

图 12 手动安装 USB3.0 驱动

（2）VC++运行库安装

安装程序将自动检测系统中是否已安装所需版本的 VC++ 运行库。若未安装或版本过低，程序将自动执行安装。

若启动软件后，出现应用程序报错或提示缺少 DLL 文件的情况，请尝试手动修复 VC++ 运行库。

请在软件安装目录下找到 vc_redist.x64.exe 文件，双击运行后，根据提示完成安装或修复（即覆盖安装）即可。

图 13 手动安装 VC++运行库

8.1.2 Linux 系统

8.1.2.1 软件安装步骤

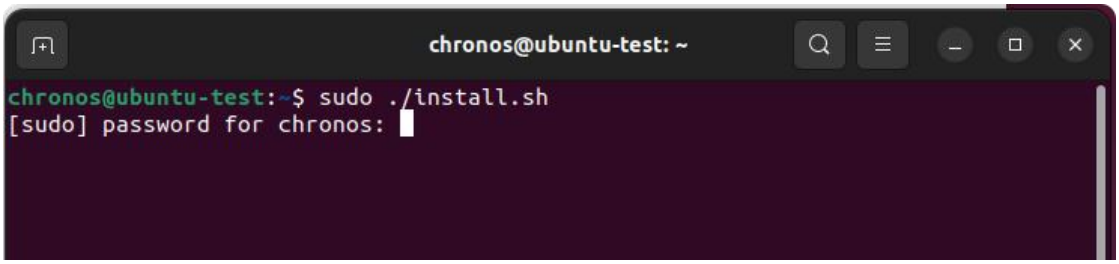
（1） 确认软件包

目前分发的软件包主要包括：

安装脚本文件：install.sh

程序压缩包文件：venus-1.0.0.tar.gz

（2） 打开终端，执行 install.sh 脚本



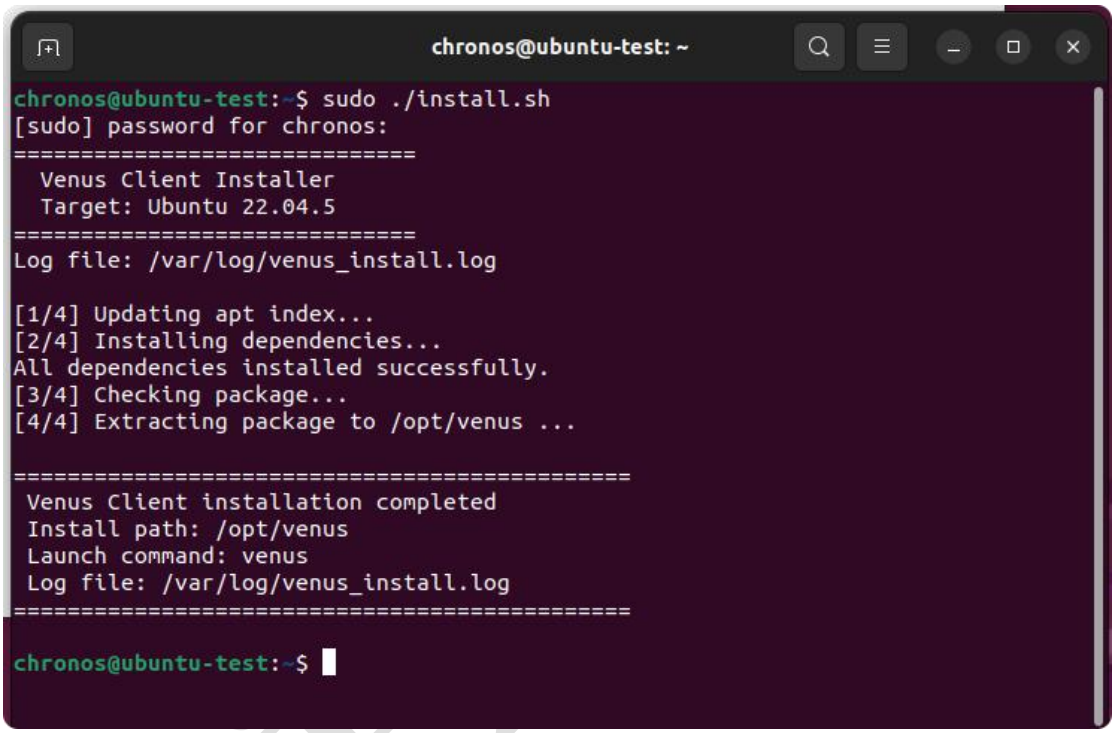


图 14 软件安装 Linux 命令

请注意，执行脚本需要 sudo 权限，输入用户密码。

安装成功安装后，会有相关提示。

（3） 软件运行

软件默认是安装在/opt/venus 目录下，可以找到可执行文件 venus，运行程序。

首先是会弹接口选择页面，选择设备接口。

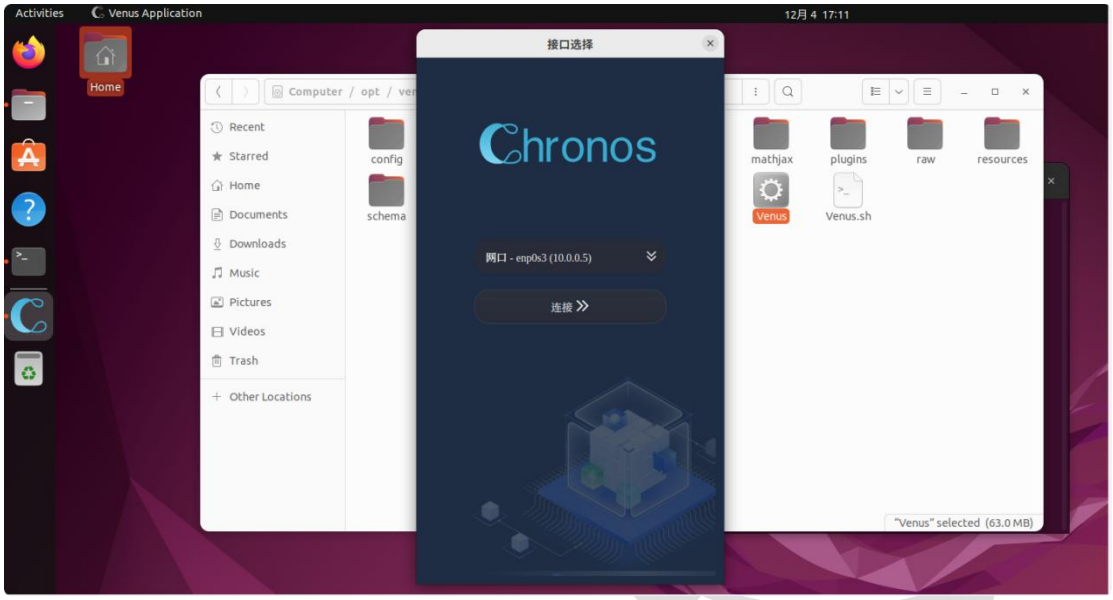


图 15 Linux 系统下的软件运行

所有的使用方法与 Windows 下相同。如果遇到问题，请联系时溯科技。

8.2 上位机电脑硬件要求

上位机软件需安装于运行 Windows 或 Linux 操作系统的个人电脑上。为充分发挥系统性能，计算机需满足以下软硬件要求。

表格 4 上位机电脑配置需求

	最低配置	推荐配置
操作系统	Windows 11 Ubuntu 22.04 LTS	Windows 11 Ubuntu 22.04 LTS
处理器	Intel：i5 4 核 8 线程 主频 1.6GHz AMD：Ryzen3 6 核 12 线程 主频 2.4GHz	Intel：13th i7 16 核 24 线程 主频 2.1GHz AMD：Ryzen7 8 核 16 线程 主频 3.8GHz
内存	16G DDR3	64G DDR5
硬盘	100G SSD + 500G HDD	500G SSD + 1T HDD
USB 端口	USB3.0	USB3.0
显示器分辨率	分辨率 1920 * 1200	分辨率 1920 * 1200
万兆网卡（选配）	/	10Gtek：X710-DA4 Intel: X722-DA4
千兆网线	符合 CAT6 标准	符合 CAT7 标准

8.3 接口配置

Venus 提供三种对外数据接口：千兆网口、万兆网口和 USB 3.0。用户可根据不

同的应用场景与性能需求选择任一接口，所有接口的用户操作流程一致。

以下以 Windows 系统为例进行说明。

8.3.1 千兆网口

Venus 设备出厂默认配置如下：

- 设备本地 IP 地址: 10.0.0.10
- 上位机电脑 IP 地址: 10.0.0.5
- 设备本地 MAC 地址: 12-34-56-78-90-AB
- 通信端口号: 1234 (设备与上位机默认使用相同端口)

请按以下步骤建立千兆网连接：

1) 使用千兆网线将 Venus 设备的千兆网口与上位机电脑的千兆网口相连；
2) 打开电脑系统的“以太网属性”对话框，选择“Internet 协议版本 4 (TCP/IPv4)”属性，将上位机电脑的 IP 地址手动设置为 10.0.0.5；
3) Venus 设备上电后，等待约 5 秒钟。连接正常建立后，可在上位机电脑的“网络连接状态”中看到“速度: 1.0 Gbps”的提示；
4) 如需永久修改 IP 地址或端口号，可通过上位机软件的“管理界面”进行配置。新的设备地址和端口信息将保存在设备内部，下次开机时无需重新设置。
5) **重要：** 修改设备 IP 地址后，必须同步更新上位机软件配置文件。请打开软件安装目录下 ./config/tdcconfig.ini 文件，将其中的 IP 地址和端口号修改为与新设备配置一致，否则软件无法正常连接。

图 16 上位机电脑千兆网网络设置

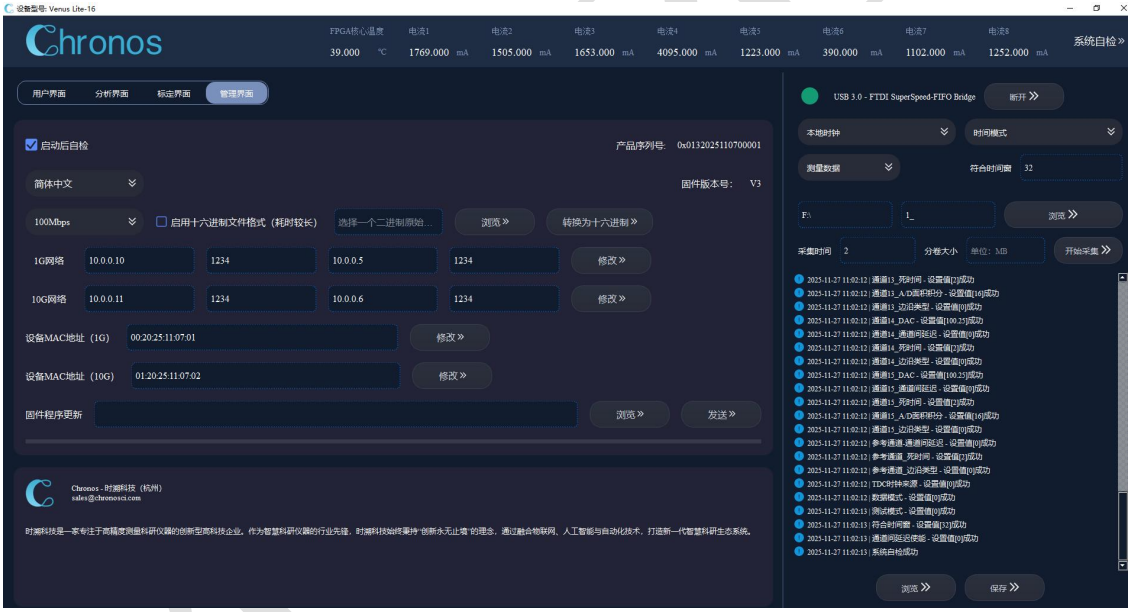


图 17 网口 IP 和端口地址修改页面

8.3.2 万兆网口

Venus 设备出厂默认配置如下：

- 设备本地 IP 地址: 10.0.0.11
- 上位机电脑 IP 地址: 10.0.0.6
- 设备本地 MAC 地址: 12-34-56-78-90-AB
- 通信端口号: 1234 (设备与上位机默认使用相同端口)

请按以下步骤建立万兆网连接：

1) 安装万兆网卡到上位机电脑，如 Intel 的 X722-DA4，并正确安装驱动程序；
2) 使用 QSFP 转 SFP 接口，或者 QSFP 一分四路 SFP 接口带光纤。其中，第一路为默认万兆网有效 SFP 接口；
3) 打开电脑系统的“以太网属性”对话框，选择“Internet 协议版本 4 (TCP/IPv4)”属性，将上位机电脑的 IP 地址手动设置为 10.0.0.6；
4) Venus 设备上电后，等待约 5 秒钟。连接正常建立后，可在上位机电脑的“网络连接状态”中看到“速度: 10.0 Gbps”的提示；
5) 如需永久修改 IP 地址或端口号，可通过上位机软件的“管理界面”进行配置。新的设备地址和端口信息将保存在设备内部，下次开机时无需重新设置。

6) **重要：** 修改设备 IP 地址后，必须同步更新上位机软件配置文件。请打开软件安装目录下 ./config/tdcconfig.ini 文件，将其中的 IP 地址和端口号修改为与新设备配置一致，否则软件无法正常连接。

图 18 万兆网物理连接示意图

图 19 上位机电脑万兆网网络设置

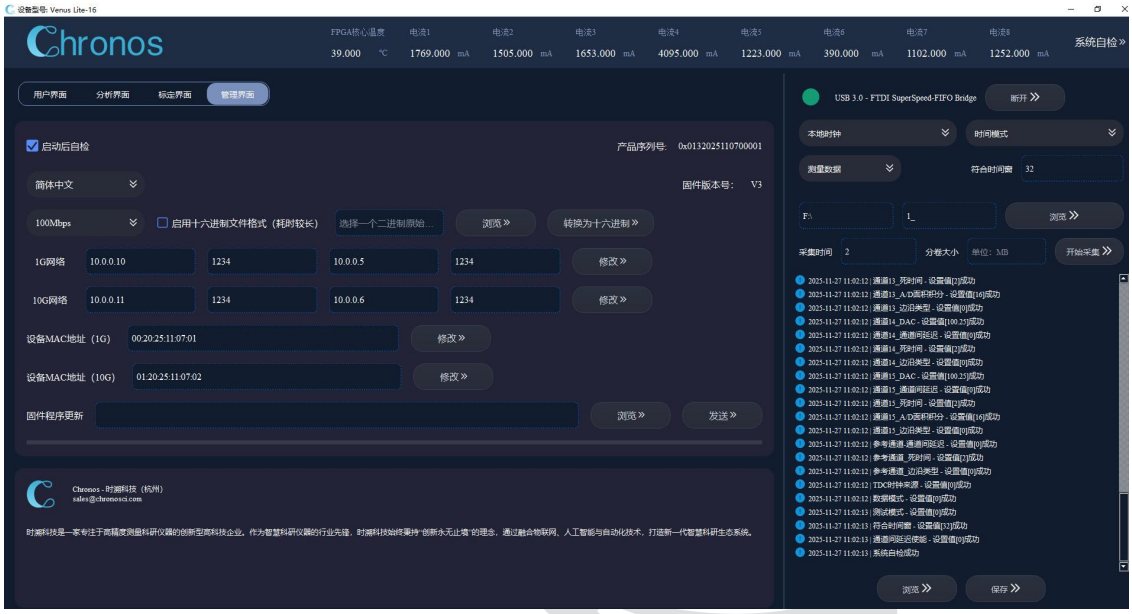


图 20 网口 IP 和端口地址修改页面

对于千兆网和万兆网，还可以通过交换机进行组网。具体的搭建过程可参考 11.2 节。

8.3.3 USB 3.0

用户需要按照以下步骤建立 USB 连接：

1) 使用 USB 线缆将 Venus 设备与上位机电脑连接；
2) 启动设备后，上位机电脑需安装 FTDI USB 驱动，驱动一般会在安装上位机软件中自动安装，如果无法识别，请按照 8.1.1.2 节中的说明进行手动安装。安装成功后，驱动信息显示如下所示。

图 21 USB3.0 驱动安装信息

8.4 界面介绍

（1） Venus 上位机软件界面主要分为以下五个功能模块：登录页面、用户页面、分析页面、系统标定页面和管理页面；

（2） 登录页面：该页面显示所有已成功连接的硬件接口。软件将自动检测 USB 3.0、千兆网及万兆网接口的连接状态，仅将连接正常的接口显示于列表中。用户可根据需要从中选择任一接口进行后续操作；

（3） 用户页面：该页面提供数据采集相关的各项参数配置选项及原始数据采集功能；

（4） 分析页面：该页面用于在线对采集的原始数据进行实时分析，使用户能够快速观测测试结果。此页面的功能会持续更新迭代；

（5） 系统标定页面：该页面用于对设备模拟前端的基线（Baseline）和噪声性能进行标定与校准；

（6） 管理页面：该页面提供辅助功能，如软件语言选择、保存数据格式、设备以太网地址配置、下位机固件在线更新等。

8.5 登录页面

图 22 登录页面

登录页面将显示所有已成功连接的接口驱动。软件会自动检测 USB 3.0、UDP1G (千兆网) 和 UDP10G (万兆网) 接口的状态，并将连接正常的接口列于表中。用户可根据需要从中选择其一，点击“连接”按钮后即可进入软件主页面（用户页面）。

8.6 用户页面

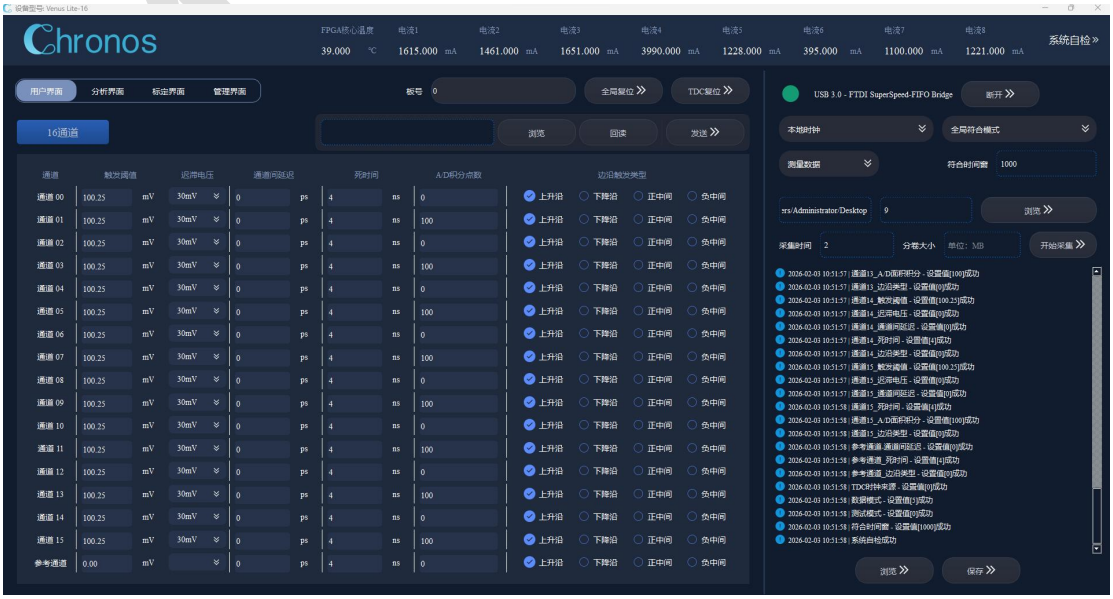


图 23 用户页面

用户页面主要包含以下四个功能区：

（1）状态监控区：位于页面上方，实时显示系统关键状态参数，包括：FPGA 核心温度、各供电支路电流值、硬件接口连接状态等；

（2）参数配置区：提供核心寄存器配置功能，主要包括

（2.1）通道配置： 通道间延迟对齐使能、迟滞电压、各通道时间甄别阈值、时间延迟值、测量死时间、ADC 积分时间、测量信号边沿设置等。以 Venus Lite16 为例，通道分为 16 路测量通道和 1 路参考通道，其中参考通道没有模拟前端电路，直接输入数字脉冲，除此之外，其他测量与普通通道无异；

（2.2）系统配置: TDC 时钟源选择、数据采集模式设置、符合时间窗设置等。

（3）数据采集区：用于控制数据采集任务，可设置数据文件的保存名称、存储路径、采集时长、分卷大小等参数；

（4）操作日志区：记录用户的所有操作与系统事件，用户可保存日志文件以备后续查阅，用于追踪和复现实验过程。

注意：如果因为意外或者测试人员造成的设备与上位机电脑通讯中断，如果设备没有重新上下电，重新建立连接后上位机软件会静默回读当前设备的所有当前状态信息。因此，仅通信重新连接不会造成下位机与上位机状态不一致。

8.6.1 比较阈值配置

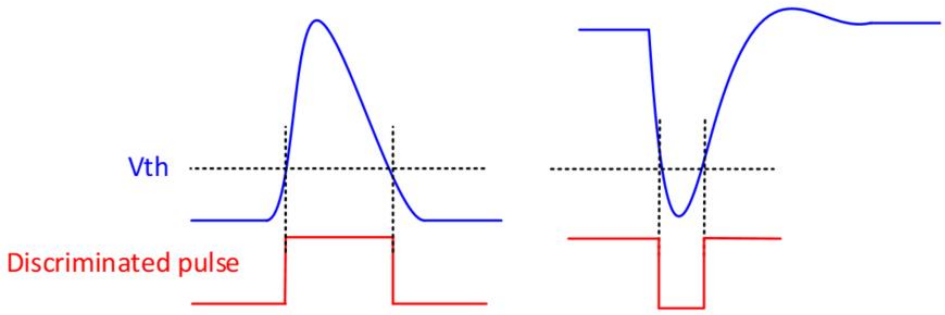


图 24 模拟信号比较功能示意图

Venus 模拟前端采用上升沿定时（Leading Edge Timing）电路。用户可通过

上位机软件页面，以 0.6 mV 的分辨率配置各通道比较器的阈值电压（单位：mV），配置范围为-2000 mV ~ +2000 mV。输入模拟信号经比较器甄别后，被转换为数字脉冲。Venus 将计算脉冲的上升沿与下降沿触发时间（时间戳），并可在线计算脉冲的过阈时间（Time over Threshold, TOT）。

为准确设置触发阈值，官方建议在首次使用设备或外界环境（如温度）发生显著变化时，先执行一次系统标定（详见第 8.8 节），然后再配置阈值。设备出厂前已默认完成此项操作，用户亦可自行标定。

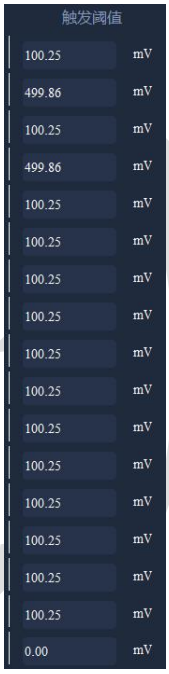


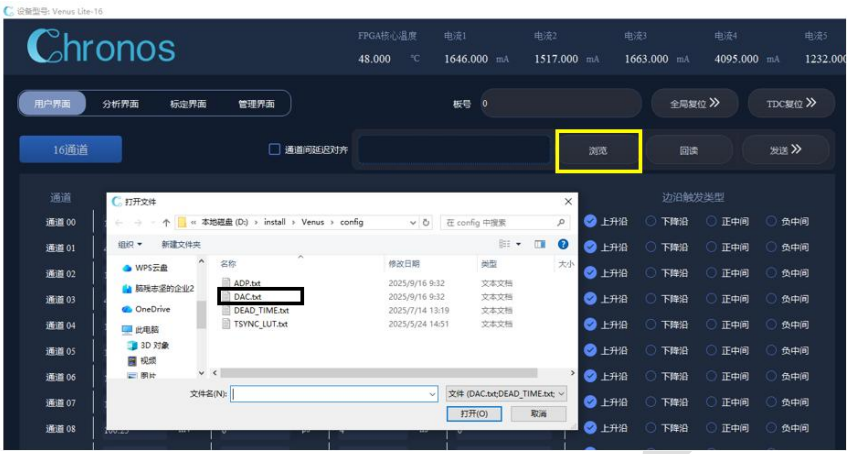
图 25 触发阈值配置 GUI 示意图

用户可通过上位机软件页面修改阈值，也可采用导入配置文件的方式快速完成批量配置。通过文件配置的操作步骤如下：

（1） 加载文件： 点击用户页面上方的“浏览”按钮，导航至 ./config 目录，选择 DAC.txt 配置文件。该文件左侧列为通道号，右侧列为对应的阈值电压值（单位：mV）；

（2） 发送配置： 点击“发送”按钮，上位机软件将自动读取文件内容并完成所有通道的阈值配置；

（3） 文件格式： DAC.txt 文件格式如下所示。用户可根据实验需求，直接修改此文件中的阈值数值；



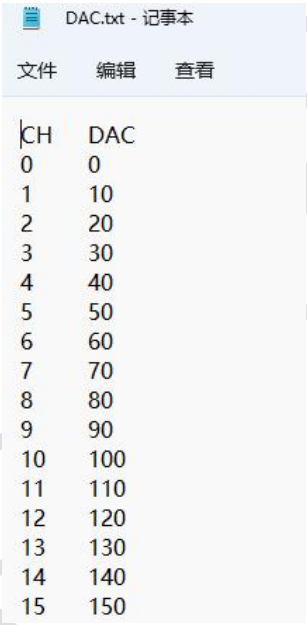


图 26 DAC.txt 文件格式要求（左侧列：通道号；右侧列：阈值电压，mV）

用户修改阈值后，改动的编辑框颜色会发生变化，配置完成后（点击回车键或者发送按钮），颜色回归正常。其他的配置也类似。

8.6.2 各通道迟滞电压配置

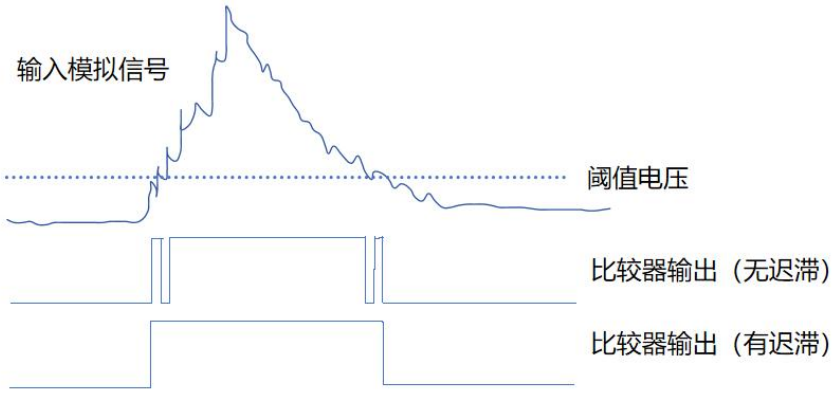


图 27 迟滞电压原理示意图

Venus 模拟前端采用边沿定时电路。用户可通过上位机软件页面，设置比较器的迟滞电压值，配置分为四个档位（1/30/40/70 mV）。输入模拟信号经比较器甄别后，被转换为数字脉冲。Venus 将计算脉冲的上升沿与下降沿触发时间（时间戳），并可在线计算脉冲的过阈时间（Time over Threshold, TOT）。如果系统噪声偏大，比较器可能在阈值电压附近反复被噪声触发，造成 TDC 编码错误和定时精度下降。通过设置迟滞电压，在某些应用中可以提高系统定时精度。一般来讲，建议先评估测试系统的噪声水平，迟滞电压设置超过系统噪声的峰峰值。

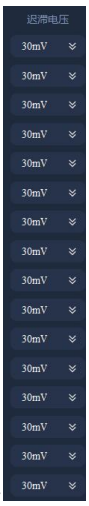


图 28 比较器迟滞电压配置 GUI 示意图

。

8.6.3 各通道延迟配置

用户可通过上位机软件配置各通道的电路延迟值，以实现通道间延迟对齐。延迟值单位为皮秒 (ps)，最小调节精度为 0.975 ps，调节范围为 1 us。

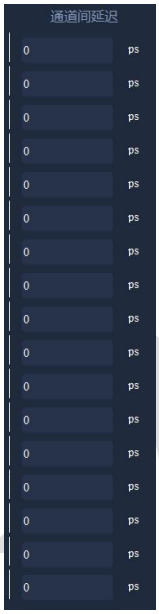


图 29 通道间延迟配置 GUI 示意图

配置延迟值可通过软件界面手动修改，也可通过导入配置文件进行批量快速配置。通过文件配置的操作步骤如下：

（1）加载配置文件： 点击用户页面上方的“浏览”按钮，导航至 ./config 目录，选择 INTER_DEALY.txt 配置文件；

（2）发送配置： 点击“发送”按钮，上位机软件将自动读取文件内容并完成所有通道的延迟配置；

（3）文件格式说明： INTER_DEALY.txt 文件格式如下所示。文件左侧列为通道号，右侧列为对应通道的电路延迟值（单位：ps，最小步进：0.975 ps）。

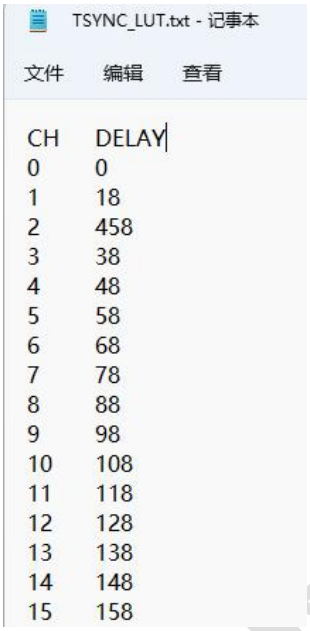


图 30 INTER_DEALY.txt 文件格式要求 （左侧列：通道号；右侧列：延迟值，ps）

8.6.4 各通道死时间配置

用户可通过上位机软件为各通道的时间测量电路配置死时间（Dead Time）。对于后沿存在“振铃”（Ringing）现象的待测信号，设置死时间可有效滤除因振铃产生的无效触发，确保数据质量。

Venus 系统的死时间模型接近非瘫痪（非扩展）型（Non-Paralyzable Model）。死时间长度由用户设定，在该时间内发生的重复触发不会再引入额外的死时间。

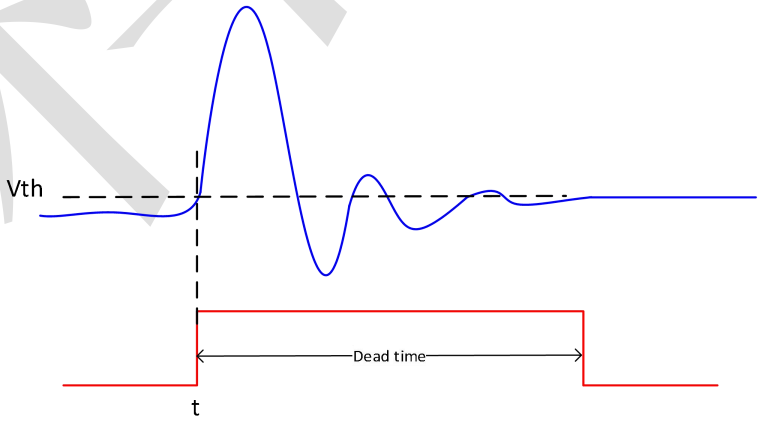


图 31 设置死时间以消除“振铃”带来的无用数据

用户可通过上位机软件界面或导入配置文件的方式，配置各通道的时间测量死时间，以实现快速批量设置。死时间单位为纳秒 (ns)，最小调节精度为 2 ns。

若用户输入值非 2 ns 的整数倍，软件将自动将其调整至最接近的合法值。默认情况下，系统测量死时间最小约 2 ns。

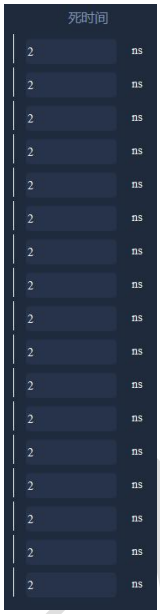


图 32 通道时间测量死时间配置 GUI 示意图

通过文件配置的操作步骤如下：

（1）加载配置文件： 点击用户页面上方的“浏览”按钮，导航至 ./config 目录，选择 DEAD_TIME.txt 配置文件；

（2）发送配置： 点击“发送”按钮，上位机软件将自动读取文件内容并完成所有通道的死时间配置；

（3）文件格式说明： DEAD_TIME.txt 文件格式如下所示。文件左侧列为通道号，右侧列为对应通道的死时间值（注意：该值必须为 2 ns 的整数倍）。

DEAD_TIME.txt - 记事本

文件 编辑 查看

CH	DEAD_TIME
0	4
1	8
2	12
3	16
4	20
5	24
6	28
7	32
8	36
9	40
10	44
11	48
12	52
13	56
14	60
15	64

图 33 DEAD_TIME.txt 文件格式要求（左侧列：通道号；右侧列：死时间，ns）

8.6.5 ADC 积分点数

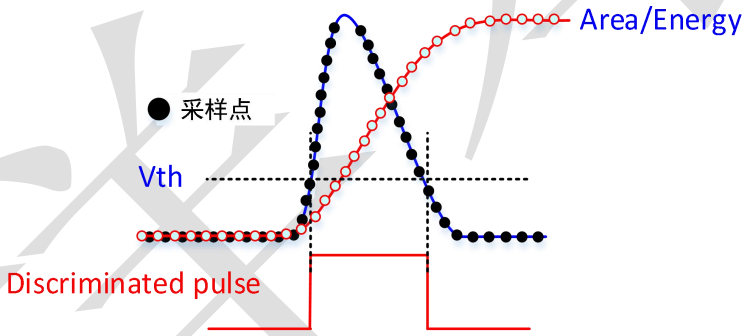


图 34 能谱测量数据模式逻辑框图

如上图所示，部分通道（标记为 AD 的通道）在过阈触发后，Venus 将启动模数转换（ADC，采样率 50 MHz，分辨率 12 位）。用户可通过上位机软件设置采样点个数，此数值即为能谱模式中的积分点数。

每次触发，Venus 会输出指定点数的幅度信息序列，用户可据此功能分析模拟信号的波形。同时，系统会将这些采样点的幅度值累加，得到信号面积（即能量信息），用户可借此功能进行能谱测量。

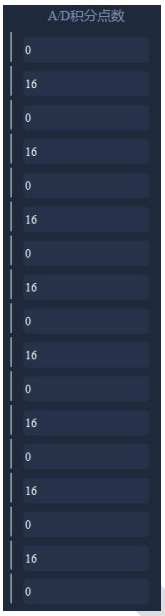


图 35 A/D 通道采样点个数配置 GUI 示意图

重要提示： Venus 系统会自动实时计算并扣除基线，以确保测量准确性。

用户可通过上位机软件界面或导入配置文件的方式，配置各通道的 ADC 积分点数。通过文件配置的操作步骤如下：

（1）加载配置文件： 点击用户页面上方的“浏览”按钮，导航至 ./config 目录，选择 ADP.txt 配置文件；

（2）发送配置： 点击“发送”按钮，上位机软件将自动读取文件内容并完成所有通道的积分点数配置；

（3）文件格式说明： ADP.txt 文件格式如下所示。文件左侧列为通道号，右侧列为对应通道的 ADC 积分点数。

图 36 ADP.txt 文件格式要求（左侧列：通道号；右侧列：积分点数）

8.6.6 信号边沿选择

Venus 可测量输入模拟信号的上升沿、下降沿时间戳，以及上升沿与下降沿时间平均值（记作 MID 方式）。用户可通过软件界面中的“边沿触发类型”选项卡进行配置。 选择原则取决于待测信号的极性和目标边沿：

（1）正脉冲： 若需测量脉冲上升沿（第一个边沿）的时间戳，应选择“上升沿”；

（2）负脉冲： 若需测量脉冲上升沿（第一个边沿）的时间戳，应选择“下降沿”；

（3）对于某些应用，需要测量信号上升沿时间戳与信号下降沿时间戳的中心处的时间，这种方式可以减小信号幅度或者直流基线变化对上升沿定时位置的影响（即 Time Walk 效应），如图 39 所示。

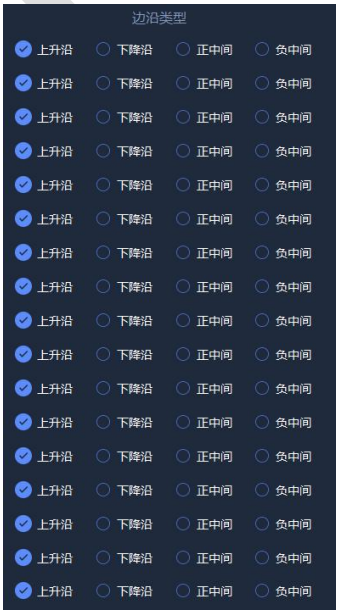


图 37 边沿采集模式配置 GUI 示意图

配置示意图如下所示。

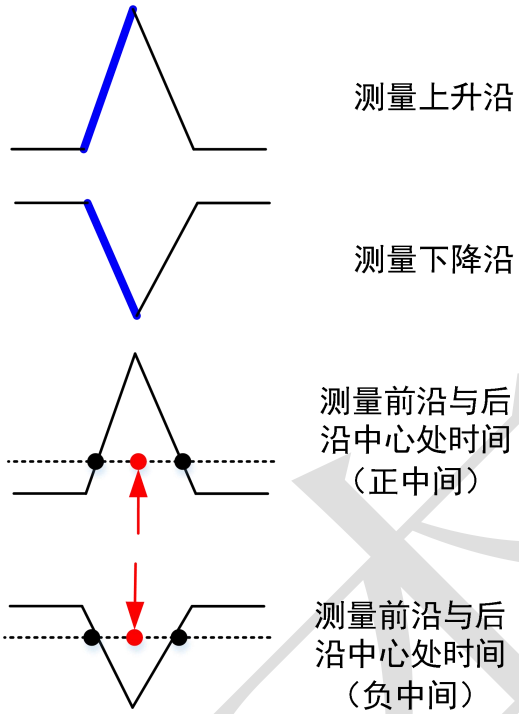


图 38 测量边沿时间戳信息选择举例（假如测量第一个边沿时间戳信息，如果输入正向脉冲，用户应选择测量上升沿；如果输入负向脉冲，用户应该选择测量下降沿。如果信号 Time Walk 或者基线漂移较大，且信号两个边沿定时性能均佳时，可以选择中间模式。）

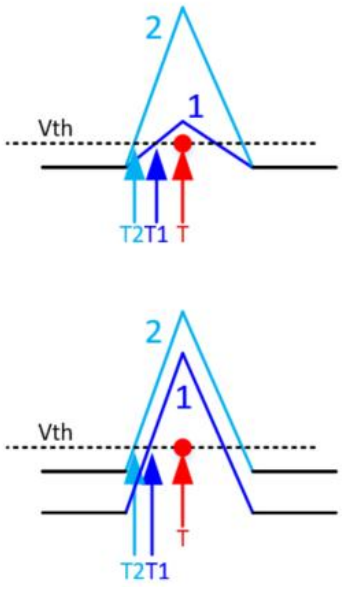


图 39 MID 测量方式举例说明（以正中间模式为例。上图：输入信号 1 幅度较小，过阈时间 T1，输入信号 2 幅度较大，过阈时间 T2。T1 与 T2 之间存在 Time Walk。采用正中间方式，对于某些信号形状固定的情形下，能够有效减小 Time Walk 的影响；下图，输入信号 1 直流基线较小，过阈时间 T1，输入信号 2 直流基线较大，过阈时间 T2。T1 与 T2 之间存在定时差。采用正中间方式，对于某些信号形状固定的情形下，能够有效减小定时差的影响。）

另外，需要注意的是，当进行 TOT 测量时，选择上升沿和下降沿会得到不同的脉宽测量结果，如下图所示。用户需要根据待测信号的特征和测量需求，选择合适的边沿测量模式。

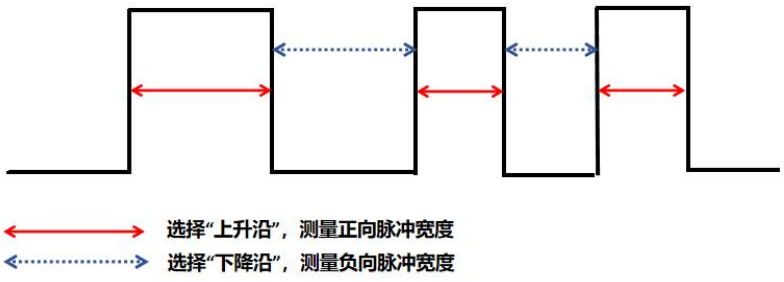


图 40 测量边沿时间戳信息选择举例（选择“上升沿”，TOT 测量中测量正向脉冲宽度；选择“下降沿”，TOT 测量中测量负向脉冲宽度。）

8.6.7 采集模式选择

系统配置需按以下步骤进行：

1. 选择时钟源

- 本地时钟：由 Venus 内部晶振产生，适用于单设备独立测试；
- 外部时钟：来自外部时钟输入接口，用于多设备组网同步测试。

2. 选择触发源

- 测量数据：使用外部输入的真实待测信号作为触发源。此为绝大多数实际应用场景下的选项；
- 内部触发：使用系统内部产生的脉冲信号作为触发源（通常用于自检或功能验证）。

3. 选择采集模式

根据测量需求，从以下模式中选择其一：

- A/D 测量数据模式
- 面积测量数据模式
- 全局符合数据模式
- 参考符合数据模式
- 过阈时间测量数据模式
- 能谱全局符合数据模式

- 双沿时间数据模式
- 周期测量数据模式

图 41 数据采集模式选择

数据采集模式解释如下：

（1） 全局符合数据模式（所有通道有效）：在此模式下，任一通道的输入信号过阈触发后，Venus 将对所有通道产生的时间戳进行在线实时排序与比较。仅当两个事件的时间差处于用户预设的符合时间窗（通过上位机软件设置，单位：ps）内时，系统才会输出一次符合事例的有效数据（包含通道号、时间差等信息）。其在线符合事例逻辑框图如下所示。具体数据格式参照 12.2 节；

在线符合逻辑图参见附录 F。

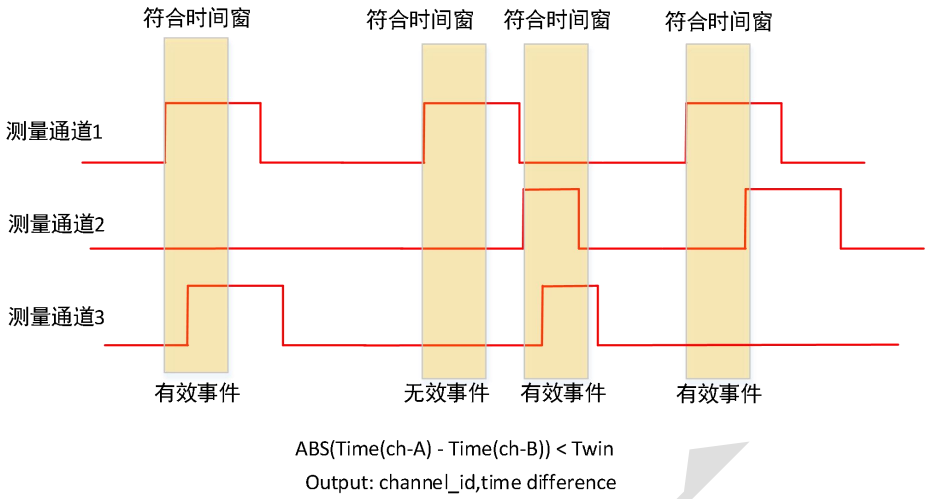


图 42 在线时间全局符合逻辑（三通道举例，实际为全通道参与符合算法）框图（测量事件边沿：上升沿）。用户设置符合时间窗，选择时间全局符合数据模式，系统会根据两个事件的时间差信息，判断是否满足符合条件。

（2） 参考符合数据模式（参考通道+所有测量通道有效）：在此模式下，任一通道的输入信号过阈触发后，Venus 将对所有通道产生的时间戳进行在线实时排序与比较。仅当两个事件的时间差处于用户预设的符合时间窗（通过上位机软件设置，单位：ps）内，且其中一个通道为参考通道时，系统才会输出一次符合事例的有效数据（包含通道号、时间差等信息）。其在线符合事例逻辑框图如下所示。具体数据格式参照 12.2 节；

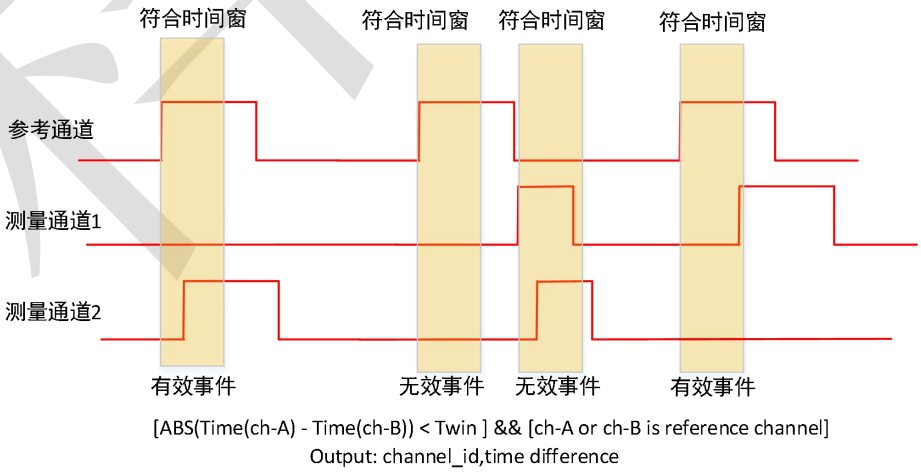


图 43 在线时间参考符合逻辑（参考通道+两个测量通道举例，实际为全通道参与符合算法）框图（测量事件边沿：上升沿）。用户设置符合时间窗，选择时间全局符合数据模式，系统会根据两个事件的时间差信息，且其中某个通道为参考通道，判断是否满足符合条件。

需要注意的是，这个模式可以看作全局符合模式的一个特例。使用全局符合模式

和离线数据筛选，可以达到相同的效果。参考电路因为没有前沿定时模拟电路，时间延迟温漂较小，此通道也可以作为标定前沿定时模拟电路温漂的一个参考。

（3） 过阈时间测量数据模式（所有通道有效）：在此模式下，通道过阈值触发后，Venus 会检测信号的上升沿与下降沿时间，并实时计算前下降沿的时间差，作为信号的脉宽值（TOT width）。该测量逻辑框图如下所示。具体数据格式参照 12.5 节；

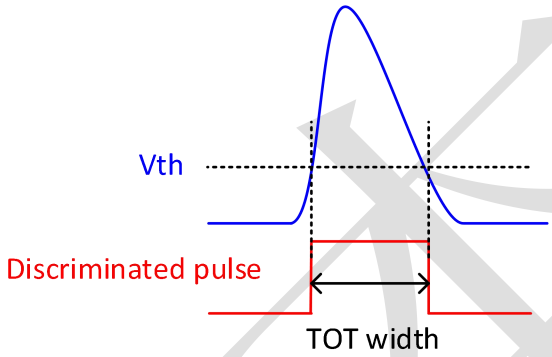


图 44 过阈时间测量逻辑框图（以正信号为例，用户应该“上升沿”边沿采集模式）

（4） ADC 测量数据模式（标注有 ADC 的通道有效）：在此模式下，当指定的通道过阈值触发后，Venus 将以 50 MHz 的采样率和 12 位的分辨率对信号进行模数转换。用户可通过上位机软件设置采样点数（该点数也对应于能谱模式中的积分门宽）。此功能可用于观测模拟信号的波形。该功能的逻辑框图如下所示。具体数据格式参照 12.3 节；

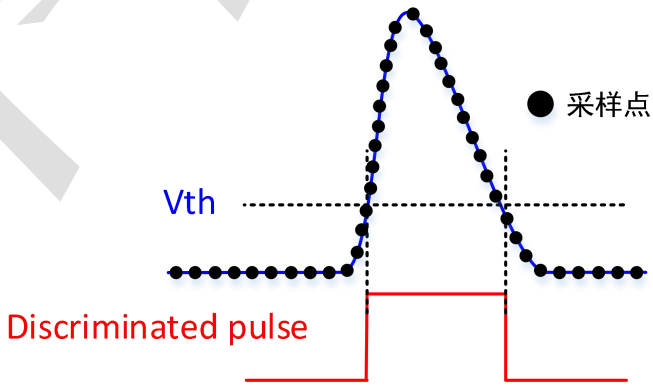


图 45 ADC 测量数据模式逻辑框图

（5） 面积测量数据模式（标注有 ADC 的通道有效）：在此模式下，当指定的通道过阈值触发后，Venus 将以 50 MHz 的采样率和 12 位的分辨率对信号

进行模数转换。用户可通过上位机软件设置采样点数（此数值即为能谱测量的积分门宽）。Venus 将对门宽内的采样点进行累加（积分），所得的面积值即代表了信号的面积（能量）信息。此功能主要用于测量输入信号的能谱。其逻辑框图如下所示。具体数据格式参照 12.4 节；

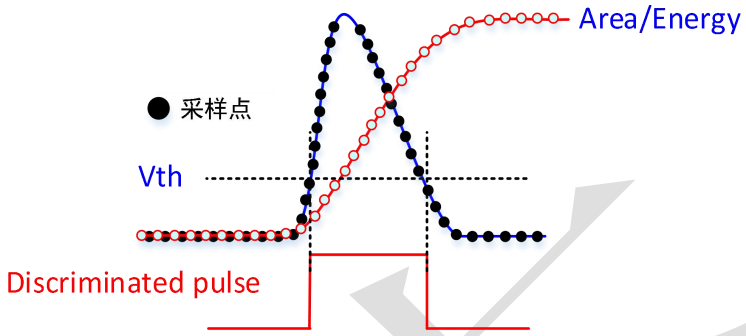
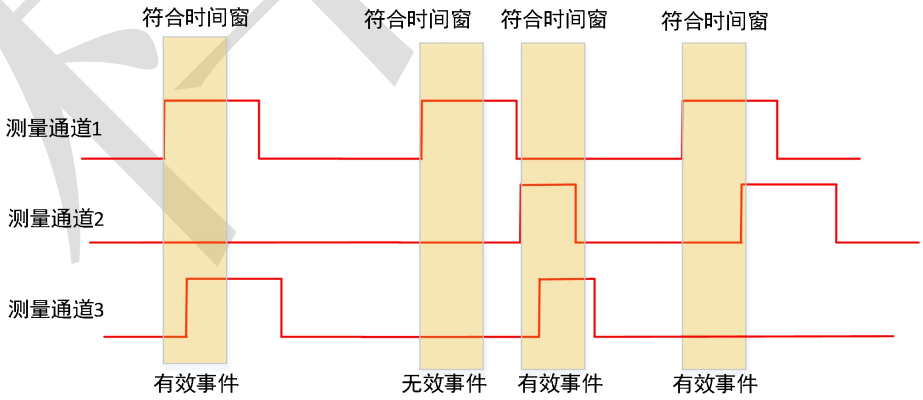


图 46 能谱测量数据模式逻辑框图

（6） 能谱全局符合测量数据模式（标注有 ADC 的通道有效）：在此模式下，当指定的通道发生过阈值触发后，Venus 会对信号的定时信息进行实时比较。当两个通道事件的时间差处于用户通过上位机软件设置的时间窗（单位：ps）内时，才将该符合事件的有效数据输出，数据内容包括：通道号、时间差值、以及相应通道的能量信息。其在线符合测量逻辑框图如下所示。具体数据格式参照 12.6 节；



ABS(Time(ch-A) - Time(ch-B)) < Twin
Output: channel_id, energy, time difference

图 47 时间-能谱全局符合测量数据模式逻辑框图 （三通道举例，实际为全通道参与符合算法）框图（测量事件边沿：上升沿）。用户设置符合时间窗，选择时间全局符合数据模式，系统会根据两个事件的时间差信息，判断是否满足符合条件。

此外，用户还可以在软件上配置能量窗，只有能量落入能窗范围内的数据才为有效数据。此模式机理和全局符合模式相同，区别仅仅是此模式输出了符合通道的面积/能量信息，在需要能量信息的应用场景（如 PET 符合探测器）中有较大的意义。

（7） 双沿时间戳数据模式（所有通道有效）：在此模式下，当通道过阈值触发后，Venus 会将信号的上升沿与下降沿的时间戳信息一并输出。用户可据此分析两个边沿的时间戳等信息，以用于分析、计算和事件筛选。时间戳的起始时刻为 TDC reset 或者外部 Sync 信号输入时刻。具体数据格式参照 12.7 节；

（8） 周期测量数据模式（所有通道有效）：在此模式下，用户输入一个时钟信号，Venus 会计算周期信号的高电平持续时间和周期信息，用户可以计算其频率、占空比等信息。具体数据格式参照 12.8 节；

综上所述，相关采集模式选择的信息见下表所示。

表格 5 裸数据采集模式说明

采集模式	英文名称	有效通道	用户需要配置信息
时间参考符合	Ref-coins	参考通道+所有测量通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源/符合时间窗等
时间全局符合	Global-coins	所有通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源/符合时间窗等
过阈时间测量	TOT	所有通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源等
A/D 测量	A/D	标注有 ADC 的通道	时间甄别阈值/死时间/积分点个数等
面积测量	Area	标注有 ADC 的通道	时间甄别阈值/死时间/积分点个数等

能谱全局符合	Area Coins	标注有 ADC 的通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源/符合时间窗/积分点个数等
双沿时间	Dual Time	所有通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源等
周期测量	Period measurement	所有通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源等

8.6.8 测量数据保存

测量数据保存，用户需要指定数据存储的路径，文件名，采集时间和数据包分卷等。其中，采集时间单位为秒，文件名系统会在后缀中自动加入当前的系统时间，分卷大小不设置默认不分卷，最大分卷大小（存储的每个文件大小）不超过 10 GB。此外，Venus 支持原始数据格式为.bin 或者.hex，即二进制数据还是十六进制数据，用户可以在管理界面中选择。

图 48 测量数据保存配置

综上，用户界面各个输入参数的合理范围、有效通道等如下表所示。

表格 6 上位机软件用户界面各个输入参数说明

输入内容	合法范围	有效通道	说明
DAC	-2000 mV ~ +2000 mV	CH0-CH15	甄别阈值
通道间延迟	0 - 10 us	CH0-CH15、参考通道	通道延迟值
死时间	2 ns - 100 us	CH0-CH15、参考通道	时间测量死时间
A/D 积分点数	0 - 65535	A/D 通道	A/D 采用和积分点数
边沿类型	上升沿、下降沿、正中间、负中间	CH0-CH15、参考通道	测量信号边沿类型
符合时间窗	0 - 16 us	全局	单位 ps
采集时间	＞0，单位秒	全局	采集时间
分卷大小	0 - 10 GB	全局	默认不分卷

8.7 分析页面

用户分析界面作为 8.6.6 节描述的各种采集模式的结果可视化输出，方便用户快速地得到数据分析结果和当前实验正确性验证。对于用户将 Venus 集成到整个实验平台中的需求，采用 API 接口方式是更好的选择。

8.7.1 强度分析

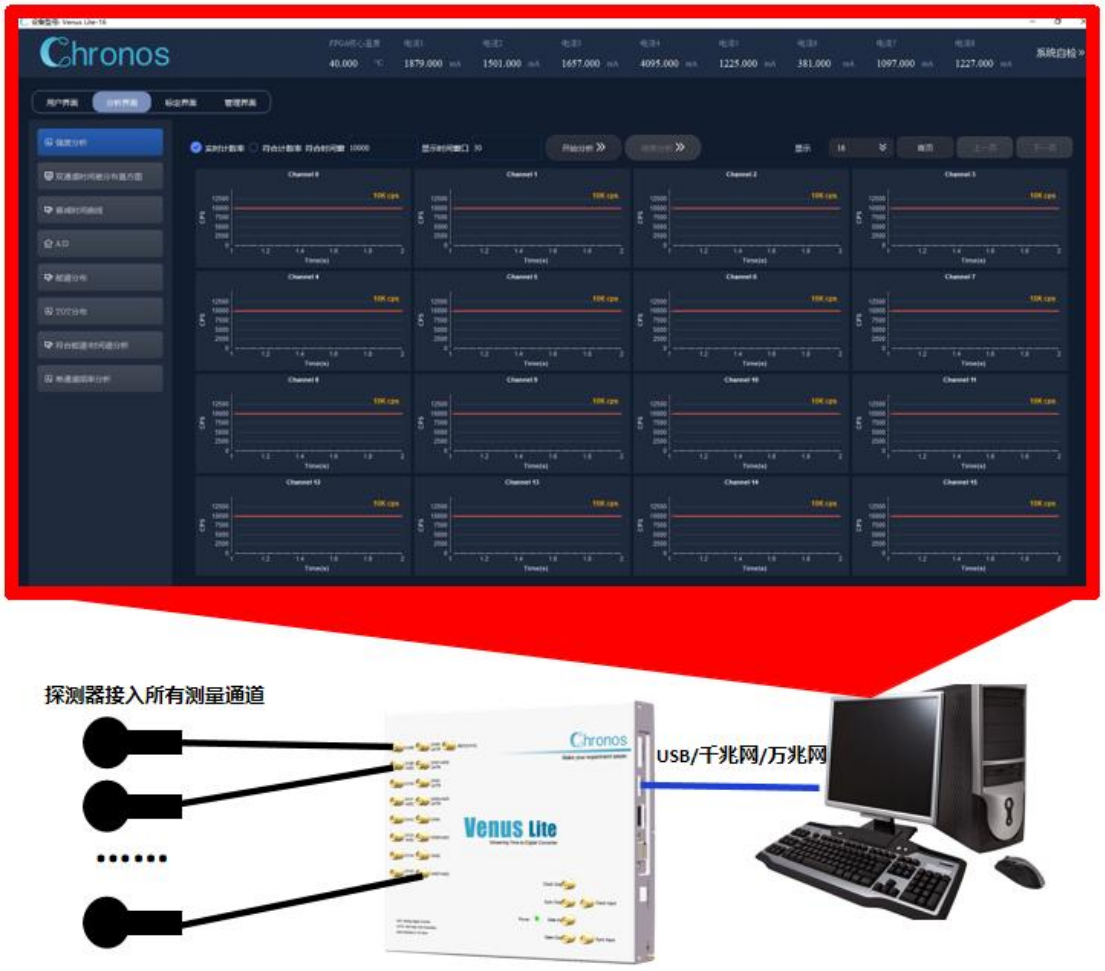


图 49 各通道强度（计数率）分析

如上图所示，用户可进入“强度分析”功能页，设定总采集时间并启动任务。启动后，Venus 会开始工作，并以每秒一次的频率刷新和显示所有通道的实时计数率（Singles Counts Per Second，SCPS）和符合计数率（Coincidence Counts Per Second，CCPS）。此功能为系统调试和性能评估提供了关键工具。

8.7.2 双通道时间差分布直方图

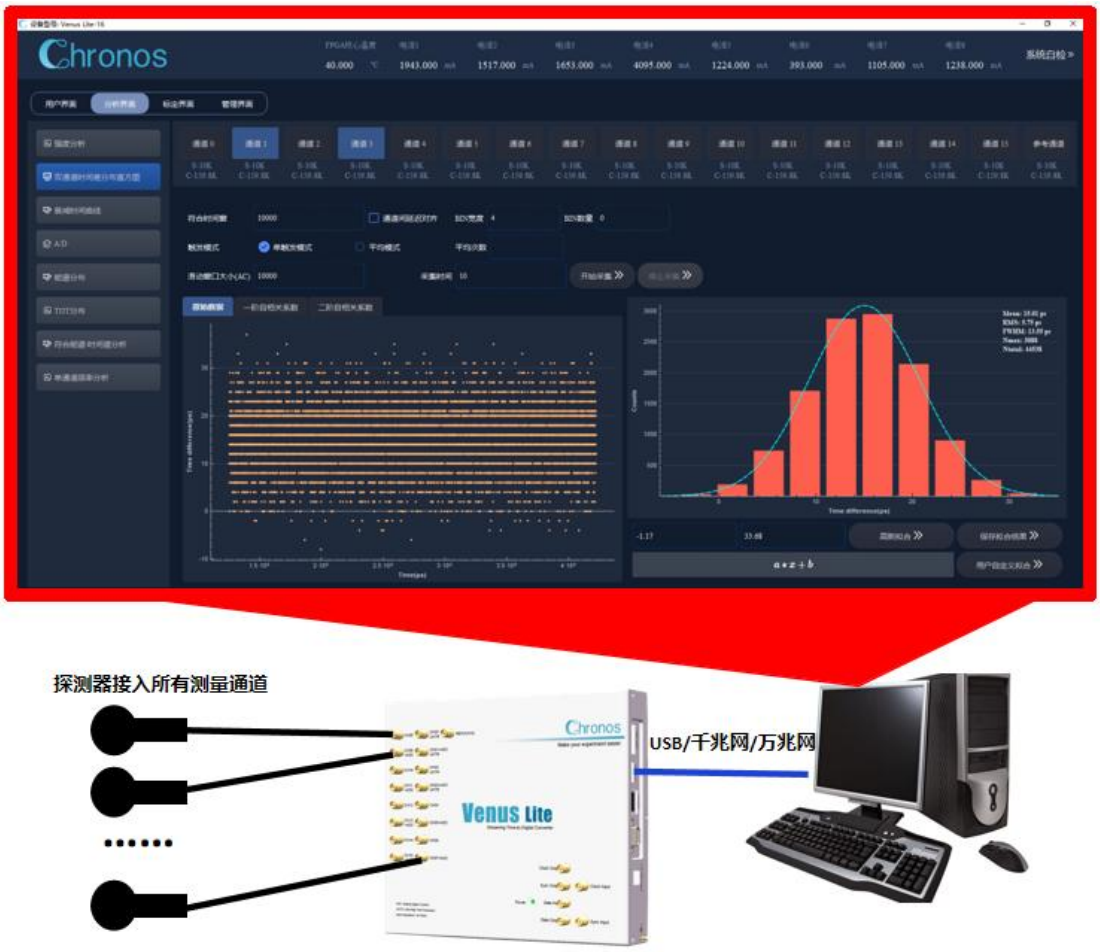


图 50 双通道时间差分布直方图

如上图所示，用户需先选择两个分析通道，并设置采集时间、符合时间窗、触发模式（单次/平均）及平均次数、直方图显示 BIN 宽等参数，随后点击“开始采集”。上位机软件会将双通道的时间差分布以直方图形式实时显示。

用户可手动输入或直接在直方图上框选拟合区间，并执行高斯拟合分析。软件将自动计算并显示拟合结果，包括：数据点数、最大值、平均值、标准差（σ）和半高全宽（FWHM）。

此外，用户还可使用自定义函数进行拟合。双击函数设置框即可弹出函数键入窗口（如下图所示），输入任意函数表达式后即可进行拟合。

图 51 自定义拟合函数键入窗口

在函数输入窗口中，根据规定的格式输入自定义函数表达式（其中常量参数用 a, b, c, d... 表示，自变量用 x 表示），随后点击“自定义拟合”按钮。上位机软件将执行拟合运算，并将得到的拟合参数（a, b, c, d...）的数值结果显示出来。所有拟合结果均可保存。

另外，Venus 还提供实时的时间差一阶和二阶归一化自相关系数结果，用户需要输入自相关系数的统计个数（即滑动窗口大小）。

此外，Venus 还实时显示各通道的单触发计数率（S）和符合计数率（C），在通道选择按钮下发实时显示。

8.7.3 Start-Stop 分组测试

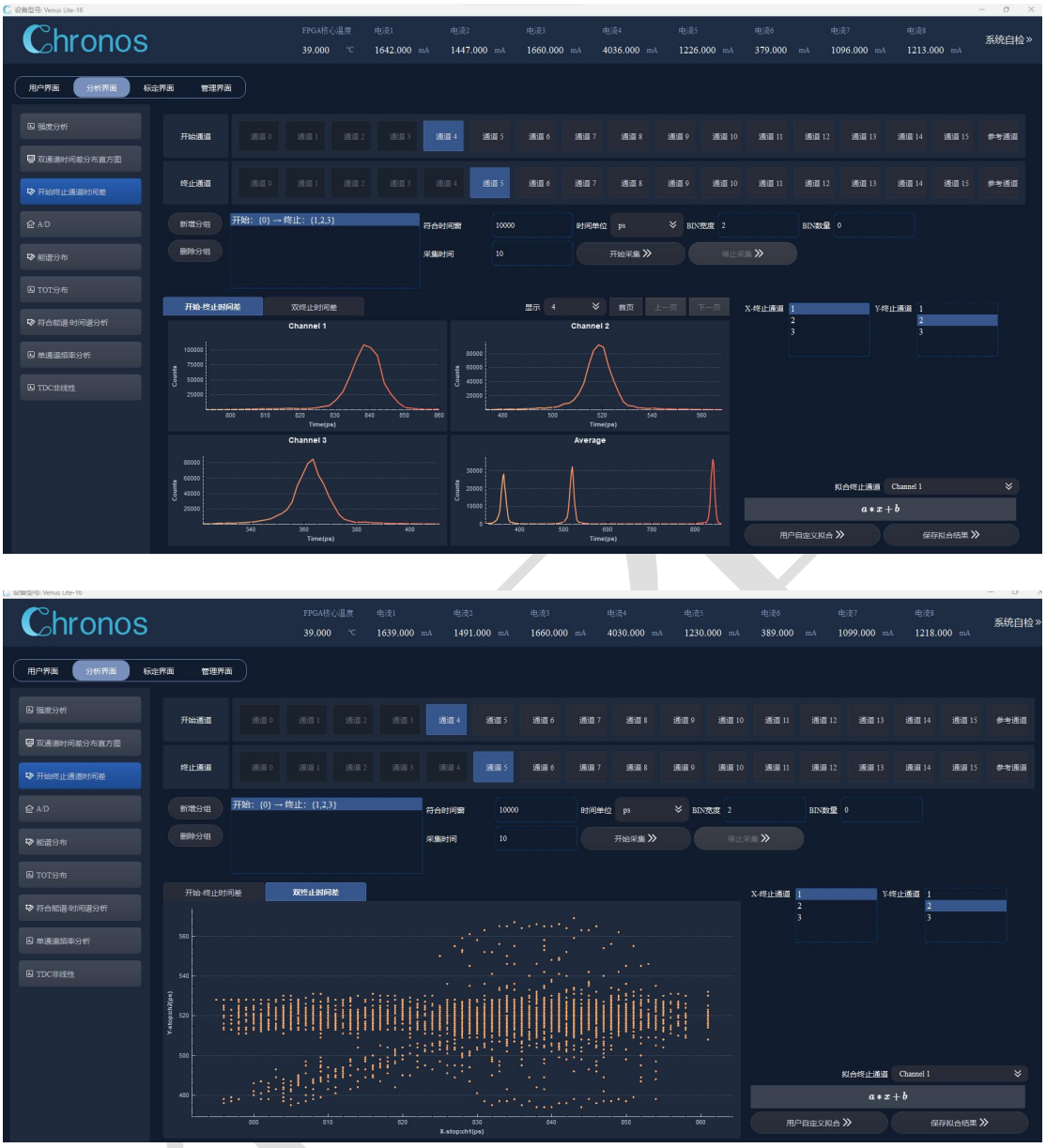


图 52 Start-stop 分组测试界面

如上图所示，在“开始和终止通道时间差”分析页面中，用户需选择一个 Start（起始）通道和多个 Stop（停止）通道，并新建分组，然后设置采集时间与符合时间窗，随后点击开始采集。上位机软件将统计所有 Stop 通道与 Start 通道的时间差，并绘制时间差的分布直方图和所有通道的平均分布（最后一张图）。该直方图反映了事件发生的时间延迟分布，可用于进一步分析如荧光寿命等物理特性。用户可以设置多个分组进行测试和分析。

另外，用户还可以选择任意 2 个终止通道，画出两个终止通道与起始通道的时间差，分别作为 X 轴和 Y 轴并画出二维 Stop 时间差图像。

8.7.4 ADC 波形

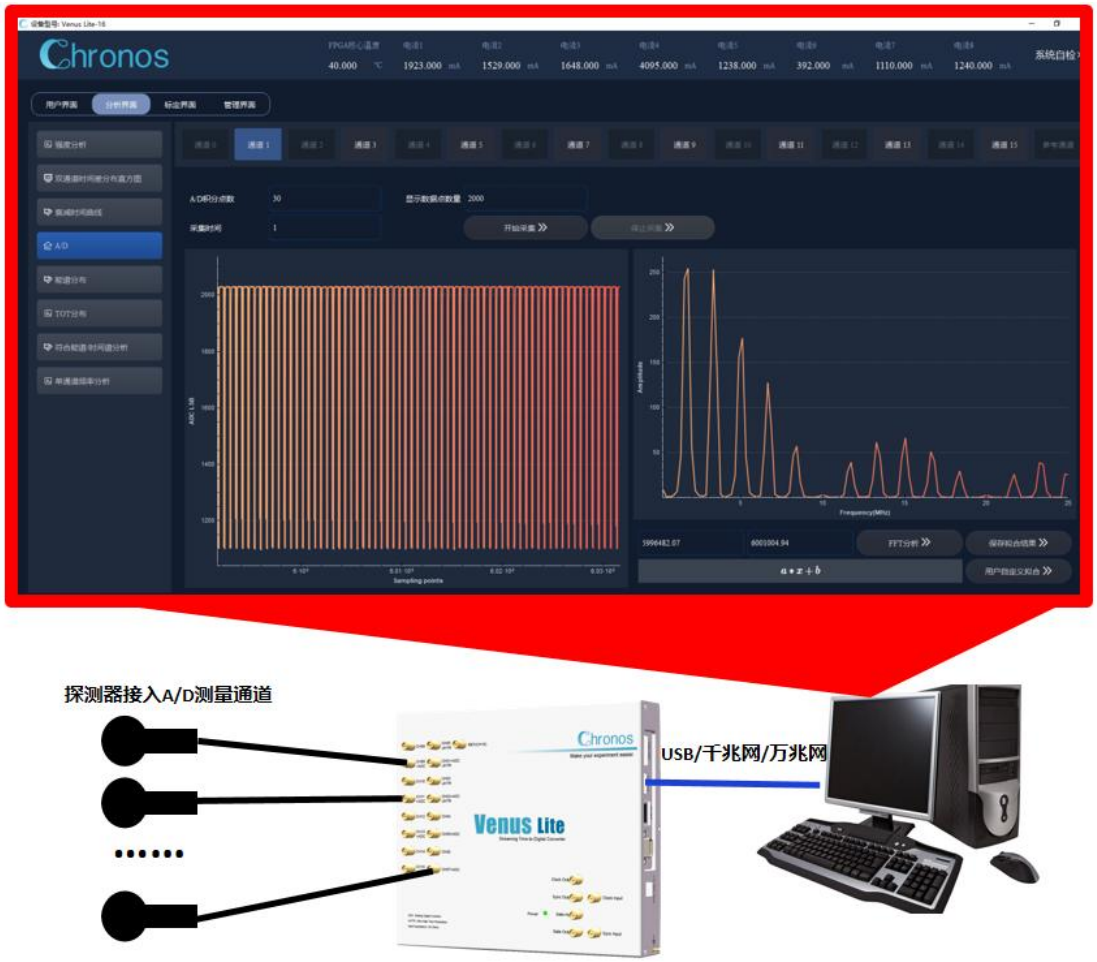


图 53 ADC 瞬态波形分析

如上图所示，用户需选择一个 A/D 输入通道，并设置采集时间和采样点数（记录长度），随后点击开始采集。上位机软件将采集到的信号数字化，并同步显示其原始波形（瞬态波形）和幅值（ADC 值）。注意：软件显示与输入信号极性反向，即输入正脉冲，显示为负脉冲。

用户可在波形图上手动输入或直接框选一个时间区间，并对该区间内的数据进行快速傅里叶变换（FFT）分析，以获取其频域特性。

8.7.5 能谱分布

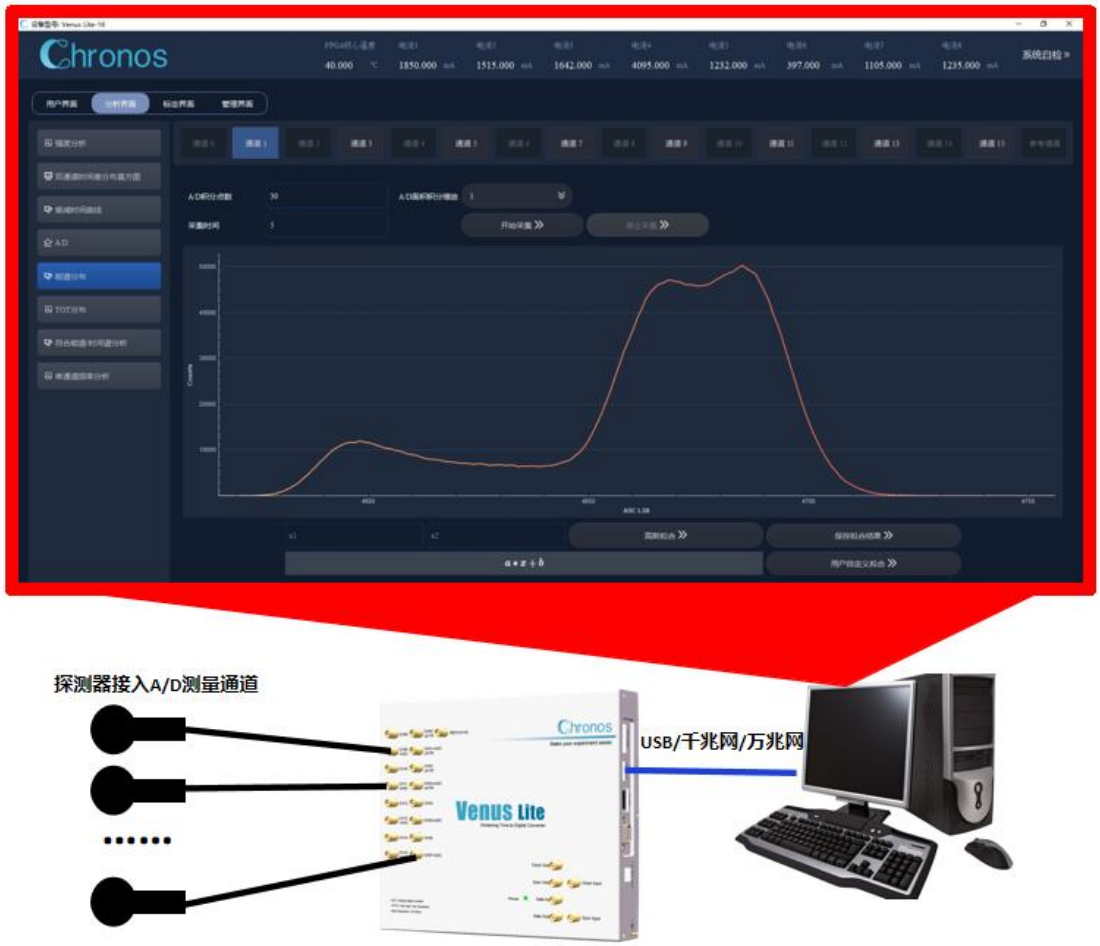


图 54 能谱分布分析

如上图所示，用户需选择一个 A/D 输入通道，并设置总采集时间和积分门宽（成形时间），随后点击开始采集。上位机软件将采集到的脉冲信号幅度（ADC 值）进行统计，最终以直方图形式显示能谱分布。

用户可选择拟合区间（手动输入道址或直接在能谱图上框选），并执行高斯拟合分析。软件将计算并显示拟合结果，包括：峰位（平均值）、峰面积（计数）、标准差（σ）和半高全宽（FWHM） 等参数。

此外，用户还可使用自定义函数进行拟合，其操作流程（选择区间后点击拟合）与上相同。

8.7.6 TOT 脉宽分布

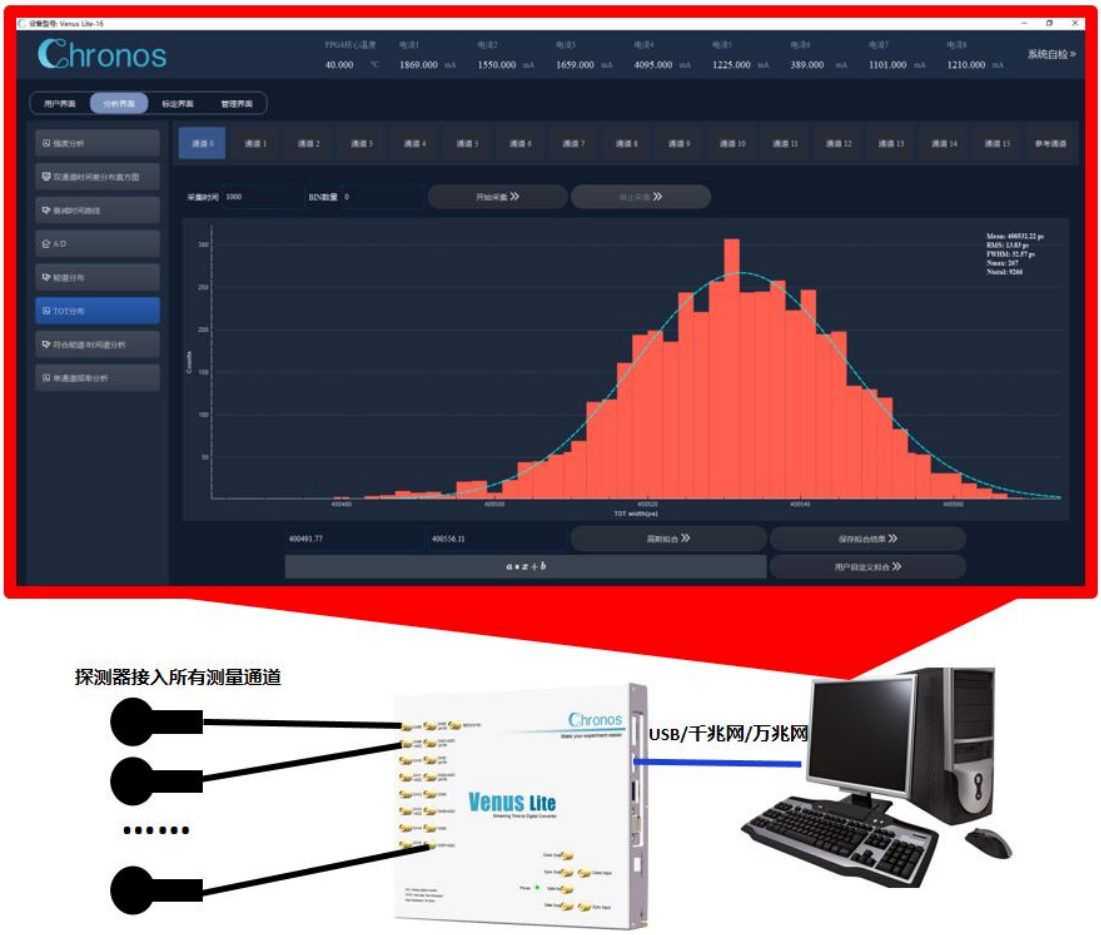


图 55 TOT 脉宽分布分析

如上图所示，用户需选择一个模拟测量通道，并设置采集时间，随后点击开始采集。上位机软件将持续测量脉冲宽度，并统计其分布，最终以直方图形式显示出来。

用户可选择拟合区间（手动输入数值或直接在直方图上框选），并执行高斯拟合分析。软件将计算并显示拟合结果，包括：总计数、最大计数值、平均脉宽、标准差（σ）和半高全宽（FWHM）。

此外，用户还可使用自定义函数进行拟合，其操作流程与上述高斯拟合一致。

8.7.7 符合能谱-时间谱分析

图 56 符合能谱-时间谱分析

如上图所示，用户需选择两个 A/D 通道，并进行基本设置，如总采集时间，触发模式（单次/平均），平均次数，符合时间窗，积分门宽（即“积分点个数”），能量有效窗。

点击开始采集后，系统将执行以下操作：

- 捕捉两个通道的时间信息；
- 筛选出时间差在符合时间窗内的符合事件；
- 从符合事件中，再筛选出能量位于能量有效窗内的事件；
- 将这些双重筛选后事件的能量值进行统计，生成并显示能谱分布直方图。

用户可选择拟合区间（手动输入或直方图上框选） 并进行高斯拟合。软件将给出拟合结果，包括：计数、最大值、峰位（平均值）、标准差（σ）和半高

全宽（FWHM）。此外，也支持自定义函数拟合，其操作方法与高斯拟合相同。

8.7.8 频率分析

图 57 单通道频率分析

如上图所示，用户需选择一个测量通道，并设置采集时间，随后点击开始采集。上位机软件将持续测量输入时钟占空比和频率，并统计周期分布，最终以直方图形式显示出来。软件会实时计算输入时钟的实时占空比、平均占空比、修正艾伦偏差（MDEV）、哈达玛偏差（HDEV）、时间偏差（TDEV）和艾伦偏差（ADEV）等信息，这些方差的计算公式见附录 G。

用户可选择拟合区间（手动输入数值或直接在直方图上框选），并执行高斯拟合分析。软件将计算并显示拟合结果，包括：总计数、最大计数值、平均脉宽、标准差（σ）和半高全宽（FWHM）。

此外，用户还可使用自定义函数进行拟合，其操作流程与上述相同。

8.8 标定页面

在系统标定界面中，当设备未连接任何外部模拟信号时，用户可设置阈值扫描范围、扫描步进（精度）和平均次数，以执行模拟前端的阈值偏置标定和噪声评估。设备将从阈值下限扫描至上限，并获取噪声触发概率曲线（计数率曲线）。对该曲线进行高斯拟合，即可得到系统的等效噪声电压和基线位置。随后，点击“自动补偿”按钮，软件将依据标定结果自动对各通道的阈值基线进行自动补偿。此功能的意义在于：消除由模拟前端电路本身引入的基线漂移，确保甄别阈值设置的准确性。

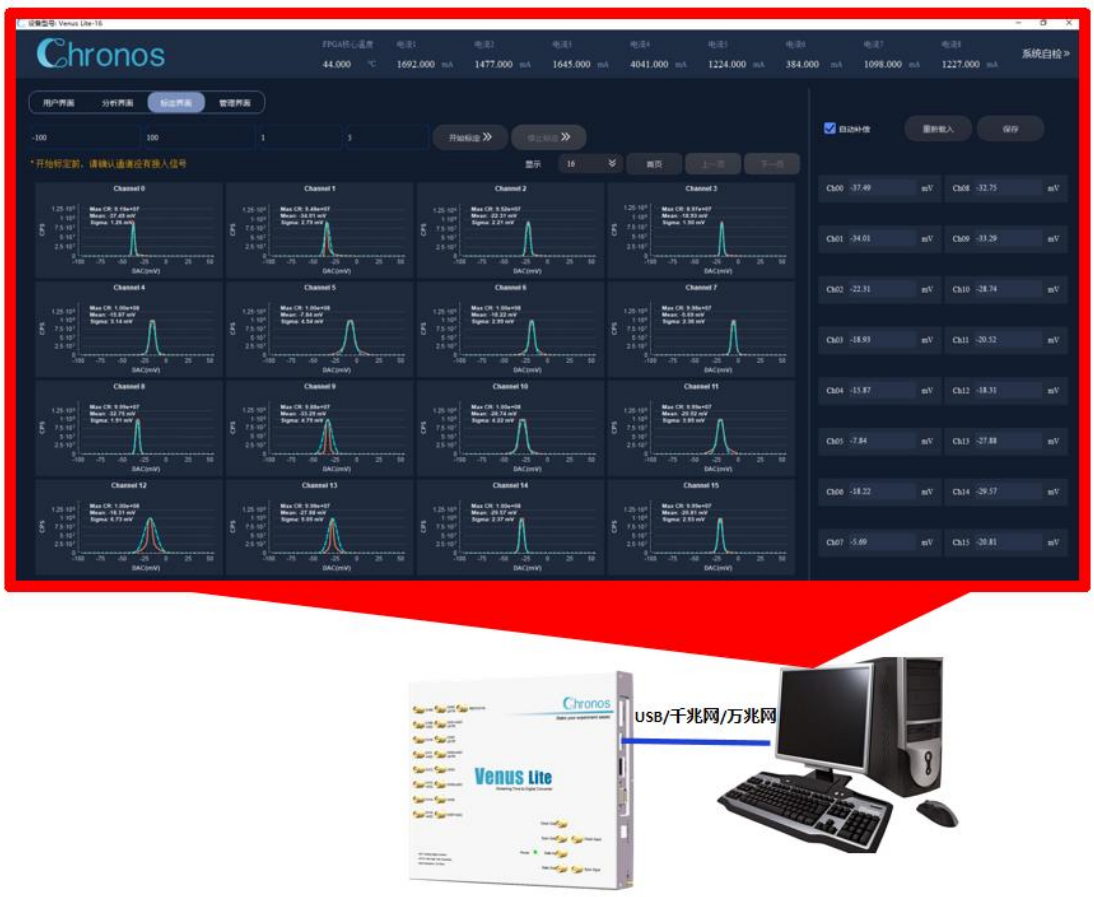


图 58 模拟前端标定

为准确设置触发阈值，建议在首次使用设备或外界环境（如温度）发生显著变化时，先执行一次系统标定，然后再设置触发阈值。标定结果温漂的性能实测

见 10.14 节。

若需补偿整个探测器系统（而不仅是设备本身）引入的基线漂移，可在相应通道接入探测器信号后进行阈值扫描与补偿。

此外，用户也可对输入信号进行阈值扫描。例如，向通道 1 和通道 3 输入固定幅度与频率的脉冲信号，便可得到如下图所示的计数率曲线。通过分析该曲线的形状特征，可以确定输入信号的幅度及其幅度的涨落。（这是一种通过阈值扫描替代 ADC 来获取能谱信息的方法）。

图 59 模拟前端标定（通道 1 和通道 3 输入固定幅度和频率信号时）

8.9 管理页面

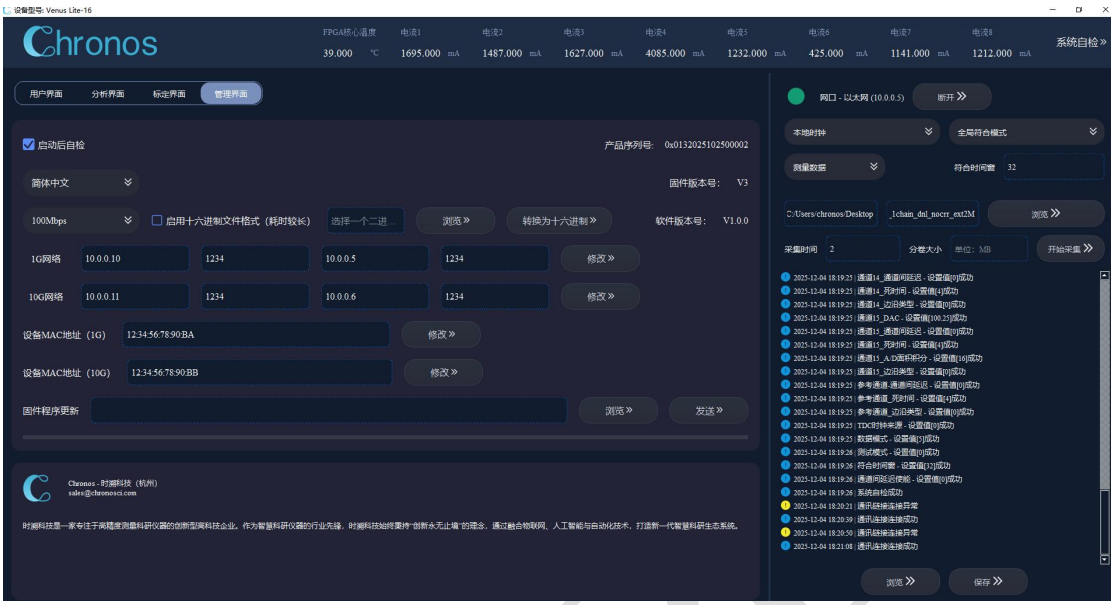


图 60 管理页面

管理页面用于进行系统级设置与信息查看，主要功能包括：

- 软件语言选择
- Raw 数据保存格式设置和离线转换
- 以太网地址配置
- 设备序列号显示
- 下位机固件版本号显示

其中，设备序列号是产品的唯一身份标识，将伴随设备的整个生命周期。凭借此序列号，我们可为每一台设备提供全生命周期的技术跟踪支持、售后服务与软件升级。

表格 7 设备序列号解释

设备序列号段	段名	可选
段 1	设备类型	0x1 Venus(TDC)
段 2	设备类型	0x1:Venus32 0x2:Venus24 0x3:Venus16lite

		0x4:Venus8lite 0x5:Venus24ultra 0x6:Venus16ultra 0x7:Venus6ultra 0x8:Venus12ultra+ 0x9:Venus8ultra+
段 3	生产日期	例如：0x20251101
段 4	生产地点	例如：0x0
段 5	生产编号	例如：0x1

8.10 日志区

图 61 系统日志

Venus 会自动保存用户的所有操作和系统时间信息，并可以保存测试过程到上位机电脑，用户可以通过管理日志来查询实验条件信息。

9. 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 device_type)	handle: 对象句柄 device_type: 设备接口类型	int - 状态码	连接指定类型设备	具体类型定义请详见 ConnType 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_ACQ_DATA_FAILED	2007	数据采集失败
CERROR_INST_PROCESS_ANALYSIS_FAILED	2008	数据分析失败
CERROR_INST_DATA_PROCESS_NOT_STOPPED	2009	上一次数据分析任务未完成
CERROR_INST_INVALID_CALIBRATION_PARAM	2010	错误的标定参数
CERROR_INST_CALIBRATION_FAILED	2011	设备标定失败
CERROR_INST_CONVERT_BINFILE_FAILED	2012	转换 BIN 文件失败

CERROR_INST_INVALID_PARAMETER	2013	错误的参数

【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	-	记录 ERROR 级别日志	

	信息			
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 设备标定

设备使用前，需要对设备进行校准处理。对各通道进行 DAC 标定处理，获取到各通道的校准值。启用自动补偿后，在后续设置各通道 DAC 时，会自动补偿校准值，确保通道 DAC 设置更为精准。

注意事项：

- 在设备标定前，请确保没有连接外部信号
- 设备标定操作耗时较长，请耐心等待
- 对同一个设备，一般只需要一次标定即可
- 为了确保完成所有通道的标定，请设置较为宽裕的起始 DAC 和终止 DAC，

比如：-100mV ~ 100mV

- 也提供手动修正通道校准值的 API
- 请注意 DAC 有效值范围：-2000mV ~ 2000mV

【API 定义】

API 函数	参数	返回值	功能描述	备注
ci_start_calibration(ci_handle_t handle, int32_t start_dac, int32_t stop_dac, double step, int32_t avg_times)	handle: 对象句柄 start_dac: 起始 DAC stop_dac: 终止 DAC step：DAC 步进值 avg_times: 平均次数	int - 状态码	设备标定	起始 DAC 要小于终止 DAC；平均次数不要过大，最大值 100
ci_set_compensation_value(ci_handle_t handle, int32_t channel, double value)	handle: 对象句柄 channel: 通道号 value: 校准值	int - 状态码	设置通道的校准值	

9.1.7 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
- 参考通道不可设置 DAC、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）

【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	-2000~2000mV
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	详见前面的 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	详见前面的 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_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_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_enable_auto_compensation(ci_handle_t handle)	handle: 对象句柄	int - 状态码	启用自动补偿	如果启用自动补偿，在设置 DAC 时，会自动加上

				特定通道的补偿值
ci_disable_auto_compensation(ci_handle_t handle)	handle: 对象句柄	int - 状态码	禁用自动补偿	

9.1.8 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-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_TIMEZONE	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

【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_10KHZ	6	10KHz 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

【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 ClockSource 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 DataMode 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 TestMode 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 TestDataBandwidth 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: 对象句柄	int - 状态码	设置符合时间窗值	范围：0~16000000ps

handle_t handle, int32_t coins_value)	coins_value: 时间窗口			
ci_set_ad_area_scale(ci_handle_t handle, int32_t scale_value)	handle: 对象句柄 scale_value: 缩放类型	int - 状态码	设置 A/D 面积积分缩放类型	详见 ADAreaScale 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: 对象句柄	int - 状态码	获取符合时间窗值	

handle_t handle, int32_t* out_value)	out_value:返回值指针			
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	

9.1.9 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 0x8: Venus Ultra Plus-8
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	int - 状态码	获取设备核心温度	

	核心温度			

9.1.10 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	int - 状态码	获取千兆网口 IP 地址

	小 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	int - 状态码	获取万兆网口端口号

	Device Port 指针		
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 size:字符数组大小	int - 状态码	获取万兆网口 MAC 地址

9.1.11 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
- 可以调用 ci_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
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_start_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	开启 RAW 数据缓存，支持实时读取 设置缓存大小，单位：MB
ci_start_rawdata_recording(ci_handle_t handle, const char* folder_path, const char* file_name, int maxVolumeSizeMB, int force_close)	handle: 对象句柄 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（默认为-1） force_close：是否强制停止文件保存（默认为 0-false）	int - 状态码	开启 RAW 数据文件保存 <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 <force_close> 如果强制停止文件保存，会立刻停止文件保存（因为异步保存文件，文件保存线程会在数据采集停止后持续运行）
ci_acq_rawdata(ci_handle_t handle, int32_t	handle: 对象句柄	int - 状态码	开始进行数据采集（异步

data_mode, int32_t duration)	data_mode: 数据模式 duration: 持续时间		模式） <data_mode> 请详见前面的 DataMode 定义 <duration> 采集时间：单位-秒
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_stop_rawdata_recording(ci_handle_t handle, int force_close)	handle: 对象句柄 force_close：是否强制停止文件保存（默认为 0-false）	int - 状态码	停止 RAW 数据保存
ci_stop_rawdata_cache(ci_handle_t handle)	handle: 对象句柄	int - 状态码	停止 RAW 数据缓存
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

9.1.12 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

【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,	handle: 对象句柄 duration: 持续时间 ch1: 通道 1 通道号	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

int32_t coins_time_window, int32_t avg_times, const char* folder_path, const char* file_name)	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			<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_fetch_tof_rawdata(ci_handle_t handle, int* 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）	int64_t*	实时读取 Tof RAW 数据	返回扁平 int64 数组，格式为：timestamp, tdiff
ci_fetch_tof_processdata(ci_handle_t handle, int* out_size)	handle：对象句柄 out_size:返回数组大小	int64_t*	实时读取 Tof Process 数据	返回扁平 int64 数组，格式为：tdiff, counts
ci_stop_tof_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 TOF 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	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

	file_name: 文件名			
ci_fetch_ad_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）	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_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	过阈值时间分析	
ci_fetch_tot_processdata(ci_handle_t handle, int* out_size)	handle: 对象句柄 out_size:返回数组大小	int64_t*	实时读取 Tot Process 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_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	int-状态码	能谱分析	<ch> 必须是 A/D 通道

	folder_path: 文件夹路径 file_name: 文件名			
ci_fetch_es_processdata(ci_handle_t handle, int* out_size)	handle: 对象句柄 out_size:返回数组大小	int64_t*	实时读取 ES Process 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 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 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 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	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_window_min2> <energy_window_max2> 能量窗的有效值范围： 0-16384

	式后的平均次数 folder_path: 文件夹路径 file_name: 文件名			
ci_fetch_conform_es_processdata(ci_handle_t handle, int* out_size)	handle: 对象句柄 out_size:返回数组大小	int64_t*	实时读取 CONFORM ES Process 数据	返回扁平 int64 数组，格式为：tdiff, counts
ci_fetch_conform_es_area1_processdata(ci_handle_t handle, int* out_size)	handle: 对象句柄 out_size:返回数组大小	int64_t*	实时读取 CONFORM ES Area1 Process 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, int* out_size)	handle: 对象句柄 out_size:返回数组大小	int64_t*	实时读取 CONFORM ES Area2 Process 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_period_analysis(ci_handle_t handle, int32_t duration,,	handle: 对象句柄 duration: 持续时间	int-状态码	周期频率分析	<sampling_interval_ms> 采样间隔，单位：ms

int32_t ch, int32_t sampling_interval_ms, int32_t sampling_num, const char* folder_path, const char* file_name)	ch: 通道 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			<sampling_num> 如果小于等于 0，全采样
ci_fetch_period_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）	int64_t*	实时读取 Period RAW 数据	返回扁平 int64 数组，格式为：time, amplitude
ci_fetch_period_processdata(ci_handle_t handle, int* out_size)	handle: 对象句柄 out_size:返回数组大小	int64_t*	实时读取 Period Process 数据	返回扁平 int64 数组，格式为：period, counts
ci_stop_period_analysis(ci_handle_t handle)	handle: 对象句柄	int-状态码	停止周期频率分析	
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	int-状态码	开始终止通道时间差分析	chan_groups 是前面定义的 cinst_chan_group_t 结构体数组指针 dual_stops_groups 是前面定义的 cinst_dual_stops_group_t 结构体数组指针

	coins_time_window: 符合时间窗值 folder_path: 文件夹路径 file_name: 文件名			
ci_fetch_startstop_processdata (ci_handle_t handle, int32_t start_chan, int32_t stop_chan, int32_t* out_size)	handle: 对象句柄 start_chan: 开始通道号 stop_chan: 终止通道号 out_size:返回数组大小	int64_t*	实时读取 Start-stop Process 数据	返回扁平 int64 数组，格式为：tdiff, counts
ci_fetch_startstop_average_processdata(ci_handle_t handle, int32_t start_chan, int32_t* out_size)	handle: 对象句柄 start_chan: 开始通道号 out_size:返回数组大小	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_size, int32_t timeout_ms)	handle: 对象句柄 start_chan: 开始通道号 out_size:返回数组大小 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_clear_analysis_data(ci_handle_t handle)	handle: 对象句柄	int-状态码	清除数据分析缓存	

9.1.13 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(char * ptr)	ptr: 内存指针	-	释放 API 分配的 Char 内存指针	主要针对 ci_fetch_rawdata 返回的内存指针，需要手动进行释放
ci_free_int64_data(char* 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_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_size)	dest: 目的内存指针 dest_size: 目的内存大小 data: 源数据地址 data_size: 源数据大小	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。

9.1.14 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_ACQ_DATA_FAILED	2007	数据采集失败
ERROR_INST_PROCESS_ANALYSIS_FAILED	2008	数据分析失败
ERROR_INST_DATA_PROCESS_NOT_STOPPED	2009	上一次数据分析任务未完成
ERROR_INST_INVALID_CALIBRATION_PARAM	2010	错误的标定参数
ERROR_INST_CALIBRATION_FAILED	2011	设备标定失败
ERROR_INST_CONVERT_BINFILE_FAILED	2012	转换 BIN 文件失败
ERROR_INST_INVALID_PARAMETER	2013	错误的参数

【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 设备标定

设备使用前，需要对设备进行校准处理。对各通道进行 DAC 标定处理，获取到各通道的校准值。启用自动补偿后，在后续设置各通道 DAC 时，会自动补偿校准值，确保通道 DAC 设置更为精准。

注意事项：

- 在设备标定前，请确保没有连接外部信号
- 设备标定操作耗时较长，请耐心等待
- 对同一个设备，一般只需要一次标定即可
- 为了确保完成所有通道的标定，请设置较为宽裕的起始 DAC 和终止 DAC，比如：-100mV ~ 100mV
- 也提供手动修正通道校准值的 API
- 请注意 DAC 有效值范围：-2000mV ~ 2000mV

【API 定义】

API	参数	参数说明	备注
bool startCalibration(int start_dac, int stop_dac, double step, int avg_times)	start_dac	起始 DAC	开始进行设备标定 返回成功或失败
	stop_dac	终止 DAC	
	step	DAC 的步进值	

	avg_times	平均次数	
bool setCompensationValue(int channel, double value)	connType	通道号	设定通道的校准值 返回成功或失败
	value	校准值	

9.2.5 通道配置

9.2.5.1 DAC 设置

可以单独设置各个通道的 DAC 值（有效范围：-2000 到 2000，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	

9.2.5.2 DAC 自动补偿设置

设备使用前，需要对各通道进行标定，获取各通道的标定值。如果启用自动补偿，在设置 DAC 时，会自动加上特定通道的补偿值。

【API 定义】

API	参数	参数说明	备注
bool enableAutoCompensation()	-	-	启动自动校准 返回成功或失败
bool disableAutoCompensation()	-	-	禁用自动校准 返回成功或失败

9.2.5.3 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>
	value	迟滞电压类型值	
INST_RESULT<int> getHysteresis(int channel)	channel	通道号	

			status - 状态 value - 迟滞电压类型值

9.2.5.4 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	

9.2.5.5 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	

9.2.5.6 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	

9.2.5.7 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>
	value	通道边沿触发类型值	
INST_RESULT<int> getEdgeType(int	channel	通道号	

channel)			status - 状态 value - 边沿触发类型值

9.2.6 全局配置

9.2.6.1 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

9.2.6.2 TDC 时钟来源设置

设置 TDC 的时钟来源：

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

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int>	srcType	时钟来源类型	设置和获取 TDC 时钟来源

setTdcClockSource(int srcType)			返回 INST_RESULT<int> status - 状态 value - 时钟来源类型值
INST_RESULT<int> getTdcClockSource()	-	-	

9.2.6.3 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_TIMEZONE	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

【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
INST_RESULT<int> getDataMode()	-	-	返回 INST_RESULT<int> status - 状态 value - 数据模式类型值

9.2.6.4 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
INST_RESULT<int> getCoinsTimeWindow()	-	-	返回 INST_RESULT<int> status - 状态 value - 符合时间窗值

9.2.6.5 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_10K_TRIGGER	6	10KHz 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()	-	-	

9.2.6.6 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()	-	-	

9.2.6.7 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
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

【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()	-	-	

9.2.6.8 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 0x8: Venus Ultra Plus-8
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;
		}

9.2.6.9 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

9.2.7 网络配置

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

9.2.7.2 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	本地网络端口	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
	device_port	设备网络端口	
INST_NET_RESULT<int> getNetPorts()	-	-	

9.2.7.3 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> setWIPs(const QString& local_ip, const QString& device_ip)	local_ip	本地 IP 地址	设置和获取万兆网 IP 返回 INST_NET_RESULT<QString> status - 状态 local_value - 本地 IP 地址
	device_ip	设备 IP 地址	
INST_NET_RESULT<QString> getWIPs()	-	-	

			device_value - 设备 IP 地址

9.2.7.4 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>
	device_port	设备网络端口	
INST_NET_RESULT<int> getWNetPorts()	-	-	status - 状态 local_value - 本地网络端口 device_value - 设备网络端口

9.2.7.5 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_MAC_RESULT getMAC()	-	-	

			INST_NET_RESULT-MAC status - 状态 device-mac - 设备网口 MAC 地址

9.2.7.6 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()	-	-	

9.2.8 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 startRawDataCache(int bufferSizeMB)	bufferSizeMB	缓存大小（单位：MB）	启动 RAW 数据缓存（支持实时从内存读取数据） 返回成功或失败
bool startRawDataRecording(const QString& folderPath, const QString& fileName, int maxVolumeSizeMB = -1, bool forceClose = false)	folderPath	输出文件目录设置（默认是 raw 目录）	启动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”。
	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（因为异步保存文	

		件，文件保存线程会在数据采集停止后持续运行）	
bool acqRawData(int dataMode, int duration)	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
	duration	数据采集时间（单位：秒）	
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 stopRawDataRecording(bool forceClose = false)	forceClose	是否强制停止文件保存	停止 RAW 数据文件保存 返回成功或失败
bool stopRawDataCache()	-	-	停止 Raw 数据缓存 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

9.2.9 数据分析

9.2.9.1 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

9.2.9.2 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进行分析。

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，或在 log 日志中进行查看
- 可以调用 stopTofAnalysis 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 stopTofAnalysis 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,
		avg_times
		}
			<ch1>
			通道 1 通道号
			<ch2>
			通道 2 通道号
			<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 fetchTofRawData(QVector<QPointF>& outData)	outData	返回 Tof Raw 数据	QPointF 格式 X-timestamp 时间戳 Y-tdiff 值

	timeout_ms	超时时间（单位：ms）	
bool fetchTofProcessData(QVector<QPointF>& outData)	outData	返回 Tof Process 数据	QPointF 格式 X-tdiff 值 Y-counts 数量
bool stopTofAnalysis()	-	-	返回成功或失败

9.2.9.3 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 stopAdAnalysis 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 stopAdAnalysis 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

9.2.9.4 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 stopTotAnalysis 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 stopTotAnalysis 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 Process 数据	QPointF 格式 X-tot 值 Y-counts 数量
bool stopTotAnalysis()	-	-	返回成功或失败

9.2.9.5 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 stopEsAnalysis 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 stopEsAnalysis 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

9.2.9.6 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 stopConformEsAnalysis 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 stopConformEsAnalysis API，停止分析

【API 定义】

API	参数	参数说明	备注
bool	duration	数据采集时间	单位：秒

conformEsAnalysis(int duration, const INST_CONFORM_ES_OPTIONS& options, const QString& folderPath, const QString& fileName)	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

9.2.9.7 周期频率分析

对指定的通道进行周期频率分析，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 stopPeriodAnalysis 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 stopPeriodAnalysis 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
		{
		ch,
		sampling_interval,
		sampling_num
		}
			<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 fetchPeriodRawData(QVector<QPointF>& outData, int timeout_ms)	outData	返回 Period Raw 数据	QPointF 格式 X-time 值 Y-amplitude 值
	timeout_ms	超时时间（单位：ms）	
bool fetchPeriodProcessData(QVector<QPointF>	outData	返回 Period Process 数据	QPointF 格式 X-period 值

& outData)			Y-counts 数量
bool stopPeriodAnalysis()	-	-	返回成功或失败

9.2.9.8 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 stopStartStopAnalysis 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 stopStartStopAnalysis 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(int start_chan, QVector<QPointF>& outData, int timeout_ms)	start_chan	开始通道号	
	outData	返回 Start-stop Dual Stops Raw 数据	QPointF 格式 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

9.2.10 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。

9.2.11 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.startRawDataCache(200);
    flag = tdc.startRawDataRecording("raw", "test");
    flag = tdc.acqRawData(INST_DATA_MODE_GLOBAL_COINS, 10);

    // 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();
    flag = tdc.stopRawDataCache();
    flag = tdc.stopRawDataRecording();

    ...
    // 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_ACQ_DATA_FAILED	2007	数据采集失败
Error.INST_PROCESS_ANALYSIS_FAILED	2008	数据分析失败
Error.INST_DATA_PROCESS_NOT_STOPPED	2009	上一次数据分析任务未完成
Error.INST_INVALID_CALIBRATION_PARAM	2010	错误的标定参数
Error.INST_CALIBRATION_FAILED	2011	设备标定失败

Error.INST_CONVERT_BINFILE_FAILED	2012	转换 BIN 文件失败
Error.INST_INVALID_PARAMETER	2013	错误的参数

【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 设备标定

设备使用前，需要对设备进行校准处理。对各通道进行 DAC 标定处理，获取到各通道的校准值。启用自动补偿后，在后续设置各通道 DAC 时，会自动补偿校准值，确保通道 DAC 设置更为精准。

注意事项：

- 在设备标定前，请确保没有连接外部信号
- 设备标定操作耗时较长，请耐心等待
- 对同一个设备，一般只需要一次标定即可
- 为了确保完成所有通道的标定，请设置较为宽裕的起始 DAC 和终止 DAC，比如：-100mV ~ 100mV
- 也提供手动修正通道校准值的 API
- 请注意 DAC 有效值范围：-2000mV ~ 2000mV

【API 定义】

API 函数	参数	返回值	功能描述	备注
start_calibration	start_dac: 起始 DAC stop_dac: 终止 DAC step：DAC 步进值 avg_times: 平均次数	bool	设备标定	起始 DAC 要小于终止 DAC； 平均次数不要过大，最大值 100
set_compensation_value	channel: 通道号 value: 校准值	bool	设置通道的校准值	

9.3.7 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
- 参考通道不可设置 DAC、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）

【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	-2000~2000mV
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
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	

enable_auto_compensation	-	bool	启用自动补偿	如果启用自动补偿，在设置 DAC 时，会自动加上特定通道的补偿值。
disable_auto_compensation	-	bool	禁用自动补偿	

9.3.8 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.TIMEZONE	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_10KHZ	6	10KHz 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

【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

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	

9.3.9 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 0x8: Venus Ultra Plus-8
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	

9.3.10 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 地址	bool	设置万兆网口 MAC 地址
get_wmac	-	str	获取万兆网口 MAC 地址

9.3.11 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
start_rawdata_cache	buffer_size_mb	bool	开启 RAW 数据缓存，支持实时读取 设置缓存大小，单位：MB
start_rawdata_recording	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）	bool	开启 RAW 数据文件保存 <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； <force_close> 如果强制停止文件保存，会立刻停止文件保存（因为异步保存文件，文件保存线程会在数据采集停止后持续运行）
acq_rawdata	data_mode: 数据模式 duration:	bool	开始进行数据采集（异步模式）

	持续时间		<data_mode> 请详见前面的 DataMode 定义 <duration> 采集时间：单位-秒
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
stop_rawdata_recording	force_close：是否强制停止文件保存（默认为 False）	bool	停止 RAW 数据保存
stop_rawdata_cache	-	bool	停止 RAW 数据缓存
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

9.3.12 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

【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: 文件夹路径 file_name: 文件名	bool	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”
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 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
fetch_tof_rawdata	timeout_ms: 超时时间（单位：ms）	List[int]	实时读取 Tof RAW 数据	返回扁平 int 数组，格式为：timestamp, tdiff
fetch_tof_processdata	-	List[int]	实时读取 Tof Process	返回扁平 int 数组，格式为：tdiff, counts

			数据	
stop_tof_analysis	-	bool	停止 TOF 分析	
ad_analysis	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	bool	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
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	
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	

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: 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	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> <energy_window_min2> <energy_window_max2> 能量窗的有效值范围： 0-16384

	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			
fetch_conform_es_processdata	-	List[int]	实时读取 CONFORM ES Process 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 CONFORM ES Area1 Process 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 CONFORM ES Area2 Process 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	
period_analysis	duration: 持续时间 ch: 通道号	bool	周期频率分析	<sampling_interval> 采样间隔，单位：ms

	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_num> 如果小于等于 0，全采样
fetch_period_rawdata	timeout_ms: 超时时间（单位：ms）	List[int]	实时读取 Period RAW 数据	返回扁平 int 数组，格式为：time, amplitude
fetch_period_processdata	-	List[int]	实时读取 Period Process 数据	返回扁平 int 数组，格式为：period, counts
stop_period_analysis	-	bool	停止周期频率分析	

startstop_analysis	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
		bool	开始终止通道时间差分析	<chan_groups: Optional[Dict[int, 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	
clear_analysis_data	-	bool	清空数据分析缓存	

9.3.13 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

9.3.14 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 calibration
res = tdc.start_calibration()
print("start_calibration->:", 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.start_rawdata_cache(200)
tdc.start_rawdata_recording("raw", "python-test")
```

```
        res = tdc.acq_rawdata(DataMode.TIMESTAMP, 10)
        print("acq_rawdata->:", res)
         raw_bytes = tdc.fetch_rawdata(500)
        time.sleep(10)
        tdc.stop_acq()

        # process data analysis
        res = tdc.tof_analysis(10, 0, 1, 10000)
        print("tof_analysis ->", res)
        data = tdc.fetch_tof_rawdata(500)
        time.sleep(10)
        tdc.stop_tof_analysis()
        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
- 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

【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_ACQ_DATA_FAILED	2007	数据采集失败
Error.INST_PROCESS_ANALYSIS_FAILED	2008	数据分析失败
Error.INST_DATA_PROCESS_NOT_STOPPED	2009	上一次数据分析任务未完成
Error.INST_INVALID_CALIBRATION_PARAM	2010	错误的标定参数
Error.INST_CALIBRATION_FAILED	2011	设备标定失败
Error.INST_CONVERT_BINFILE_FAILED	2012	转换 BIN 文件失败
Error.INST_INVALID_PARAMETER	2013	错误的参数

【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 设备标定

设备使用前，需要对设备进行校准处理。对各通道进行 DAC 标定处理，获取到各通道的校准值。如果启用自动补偿，在后续设置各通道 DAC 时，会自动补偿校准值，确保通道 DAC 设置更为精准。

注意事项：

- 在设备标定前，请确保没有连接外部信号
- 设备标定操作耗时较长，请耐心等待
- 对同一个设备，一般只需要一次标定即可
- 为了确保完成所有通道的标定，请设置较为宽裕的起始 DAC 和终止 DAC，比如：-100mV ~ 100mV
- 也提供手动修正通道校准值的 API
- 请注意 DAC 有效值范围：-2000mV ~ 2000mV

【API 定义】

API 函数	参数	返回值	功能描述	备注
startCalibration	start_dac: 起始 DAC stop_dac: 终止 DAC step：DAC 步进值 avg_times: 平均次数	logical	设备标定	起始 DAC 要小于终止 DAC； 平均次数不要过大，最大值 100
setCompensationValue	channel: 通道号 value: 校准值	logical	设置通道的校准值	

9.4.7 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
- 参考通道不可设置 DAC、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）

【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	-2000~2000mV
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	0~1000000ps
setDeadTime	channel: 通道号 value: 死时间	logical	设置通道死时间	2~130000ps
setEdgeType	channel: 通道号	logical	设置边沿触发类型	详见前面的

	value: 边沿触发类型			EdgeType 定义
setADIntegrationPoint	channel: 通道号 value: 积分点	logical	设置 A/D 面积积分值	0~65535
getDAC	channel: 通道号	double	获取通道 DAC 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	
enableAutoCompensation	-	logical	启用自动补偿	如果启用自动补偿，在设置 DAC 时，会自动加上特定通道的补偿值。
disableAutoCompensation	-	logical	禁用自动补偿	

9.4.8 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.TIMEZONE	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_10KHZ	6	10KHz 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

【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 AD 区域缩放	详见 ADAreaScale 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 AD 区域缩放	

9.4.9 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

				0x8: Venus Ultra Plus-8
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	

9.4.10 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:	logical	设置万兆网口 IP 地址

	设备 IP		
setWNetPorts	local_port: 本地端口 device_port: 设备端口	logical	设置万兆网口端口号
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

9.4.11 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
startRawDataCache	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
startRawDataRecording	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）	logical	开启 RAW 数据文件保存 <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； <force_close> 如果强制停止文件保存，会立刻停止文件保存（因为异步保存文件，文件保存线程会在数据采集停止后持续运行）
acqRawData	data_mode: 数据模式 duration: 持续时间	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
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
stopRawDataRecording	force_close：是否强制停止文件保存（默认为 False）	logical	停止 RAW 数据保存
stopRawDataCache	-	logical	停止 RAW 数据缓存
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

9.4.12 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 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	logical	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
fetchTofRawData	timeout_ms: 超时时间（单位：ms）	(success, tofData)	实时读取 Tof RAW 数据	返回扁平 int64 数组，格式为：timestamp, tdiff
fetchTofProcessData	-	(success, tofData)	实时读取 Tof Process 数据	返回扁平 int64 数组，格式为：tdiff, counts
stopTofAnalysis	-	logical	停止 TOF 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	logical	A/D 信号分析	<ch> 必须是 A/D 通道 <sampling_interval> 采样间隔，单位：ms

	间隔 sampling_num: 采样数量 folder_path: 文件夹路径 file_name: 文件名			<sampling_num> 如果小于等于 0，全采样
fetchAdRawData	timeout_ms: 超时时间（单位：ms）	(success, adData)	实时读取 A/D RAW 数据	返回扁平 int64 数组，格式为：sampling_points, adc
stopAdAnalysis	-	logical	停止 A/D 信号分析	
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	-	(success, totData)	实时读取 Tot Process 数据	返回扁平 int64 数组，格式为：tot, counts
stopTotAnalysis	-	logical	停止 TOT 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	logical	能谱分析	<ch> 必须是 A/D 通道

	径 file_name: 文件名			
fetchEsProcessData	-	(success, esData)	实时读取 ES Process 数据	返回扁平 int64 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 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	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_max1> <energy_window_min2> <energy_window_max2> 能量窗的有效值范围：0-16384

	file_name: 文件名			
fetchConformEsProcessData	-	(success, esData)	实时读取 CONFORM ES Process 数据	返回扁平 int64 数组，格式为：tdiff, counts
fetchConformEsArea1ProcessData	-	(success, esData)	实时读取 CONFORM ES Area1 Process 数据	返回扁平 int64 数组，格式为：adc, counts
fetchConformEsArea2ProcessData	-	(success, esData)	实时读取 CONFORM ES Area2 Process 数据	返回扁平 int64 数组，格式为：adc, counts
stopConformEsAnalysis	-	logical	停止符合能谱分析	
periodAnalysis	duration: 持续时间 ch: 通道号 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	周期频率分析	<sampling_interval> 采样间隔，单位：ms <sampling_num> 如果小于等于 0，全采样
fetchPeriodRawData	timeout_ms: 超时时间（单位：ms）	(success, periodDa	实时读取 Period RAW 数	返回扁平 int64 数组，格式为：time, amplitude

		ta)	据	
fetchPeriodProcessData	-	(success, periodData)	实时读取 Period Process 数据	返回扁平 int64 数组，格式为：period, counts
stopPeriodAnalysis	-	logical	停止周期频率分析	

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	(success, startStopData)	实时读取 Start-stop Process 数据	返回扁平 int64 数组，格式为：tdiff, counts
fetchStartStopAverageProcessData	start_chan: 开始通道号	(success, startStopData)	实时读取 Start-stop Average Process 数据	返回扁平 int64 数组，格式为：tdiff, counts
fetchStartStopDualStopsRawData	start_chan: 开始通道号 timeout_ms: 超时时间（单位：ms）	(success, startStopData)	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
stopStartStopAnalysis	-	logical	停止开始终止通道时间差分	

			析	
clearAnalysisData	-	logical	清空数据分析缓存	

9.4.13 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

9.4.14 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. 通道标定操作示例
fprintf('\n7. Channel calibration operations...\n');
result = tdc.startCalibration(-20, 20, 5, 3);
if result
fprintf('Process calibration successful\n');
else
fprintf('Process calibration falied\n');
end

result = tdc.setCompensationValue(0, -15.50);

%% 8. 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
.....

%% 9. 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
.....
%% 10. 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);
.....

%% 11. 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);
.....

%% 12. 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.INCREASING);
tdc.startRawDataRecording("raw", "matlab-test");
result = tdc.acqRawData(DataMode.TIMESTAMP, 10);
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();
tdc.stopRawDataRecording();

%% 13. 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	获取当前可用的设备列表	

		（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		
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、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）

【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

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 channel: 通道号 dead time: 死时间 error in: 错误输入	ChronosInst out: ChronosInst 实例 ret: 操作状态码 error out: 错误输出	设置通道死时间	2~130000ps
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	ChronosInst out: ChronosInst 实例	设置 A/D 面积积分值	范围：0~65535

	channel: 通道号 A/D integration point: A/D 积分点数 error in: 错误输入	ret: 操作状态码 error out: 错误输出		
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 out:	获取通道死	

	ChronosInst 实例 channel: 通道号 error in: 错误输入	ChronosInst 实例 ret: 操作状态码 dead time: 死时间 error out: 错误输出	时间	
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	
ciEnableAutoCompensation	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	启用自动补偿	如果启用自动补偿，在设置 DAC 时，会自动加上特定通

		出		道的补偿值。
ciDisableAutoCompensation	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	禁用自动补偿	

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
- 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_TIMEZONE	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_10KHZ	6	10KHz 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 error out: 错误输出	设置符合时间窗值	范围： 0~16000000ps
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 error in: 错误输入	ChronosInst out: ChronosInst 实例 ret: 操作状态码 test mode: 测试模式 error out: 错误输出	获取测试模式	
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	获取测试数据带宽	

		error out: 错误输出		
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: 错误输出	获取通道数量	获取设备通道数，不包含参考通道
ciGetDeviceType	ChronosInst in: ChronosInst 实例 error in: 错误输入	ChronosInst out: ChronosInst 实例 device type: 设备	获取设备类型	0x1：Venus(TDC) 0x2: Mercury(多道分析仪)

		类型 error out: 错误输出		
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 0x8: Venus Ultra Plus-8
ciGetProductSN	ChronosInst in: ChronosInst 实例 error in: 错误输入	ChronosInst out: ChronosInst 实例 product sn: 设备产品序列号 error out: 错误输	获取产品序列号	

		出		
ciGetDeviceStatus	ChronosInst in: ChronosInst 实例 error in: 错误输入	ChronosInst out: ChronosInst 实例 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

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	ChronosInst out: ChronosInst 实例 ret: 操作状态码 error out: 错误输	设置千兆网口端口号	

	1G device port: 千兆网设备网络端口 error in: 错误输入	出		
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 error out: 错误输	获取千兆网口 IP 地址	

		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	ChronosInst out: ChronosInst 实例	获取万兆网口端口号	

	error in: 错误输入	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		
ciSetMAC	ChronosInst in: ChronosInst 实例 1G device MAC: 千兆网设备 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	设置千兆网口 MAC 地址	
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	ChronosInst out: ChronosInst 实例 ret: 操作状态码	设置万兆网口 MAC 地址	

	网设备 MAC 地址 error in: 错误输入	error out: 错误输出		
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 out:	禁用十六进	

	ChronosInst 实例 error in: 错误输入	ChronosInst 实例 ret: 操作状态码 error out: 错误输出	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	
ciStartRAWDataCache	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
ciStartRAWDataRecording	ChronosInst in: ChronosInst 实例 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）	ChronosInst out: ChronosInst 实例 ret: 操作状态码 error out: 错误输出	开启 RAW 数据文件服务器端保存	<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”;

	force close：是否强制停止文件保存（false） error in: 错误输入			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； <force close> 如果强制停止文件保存，会立刻停止文件保存（因为异步保存文件，文件保存线程会在数据采集停止后持续运行）
ciStartRAWDataLocalRecording	ChronosInst in: ChronosInst 实例 file path: 数据文件目录 file name: 数据文件名 max volume	ChronosInst out: ChronosInst 实例 ret: 操作状态码 error out: 错误输出	开启 RAW 数据文件本地保存	file path 默认是 raw 目录

	size(MB)：文件分卷大小（默认为0） buffer size(MB)：本地数据缓存大小 error in: 错误输入			
ciAcqRAWData	ChronosInst in: ChronosInst 实例 duration: 数据采集持续时间 data mode: 数据模式设置 error in: 错误输入	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：单位-秒
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	ChronosInst out: ChronosInst 实例 ret: 操作状态码	停止数据采集	

	入	error out: 错误输出		
ciStopRAWDataRecording	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 RAW 数据服务端保存	
ciStopRAWDataCache	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 RAW 数据缓存	
ciConvertToHex	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	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
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	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	

	error in: 错误输入			
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	采集数据并保存到服务端文件	duration delay 默认是 2 秒
acqRAWDataToLocalFile	ChronosInst in: ChronosInst 实例 duration: 数据采	ChronosInst out: ChronosInst 实例	采集数据并保存到本地文件	duration delay 默认是 2 秒

	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） buffer size(MB)：本地数据缓存大小 error in：错误输入	ret：操作状态码 error out：错误输出		

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 out:	获取指定通道的	

	ChronosInst 实例 channel: 指定通道号 error in: 错误输入	ChronosInst 实例 single cps: 实时计数率 error out: 错误输出	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 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	输出所有通道的符合计数率到文件	<file path> 默认是 output 目录

	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
ciFetchTofRAWData	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 TOF 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：timestamp,

				tdiff
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 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 TOF 数据分析	
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	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

	file name: 数据文件名 error in: 错误输入			
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 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	<ch> 必须指定 A/D 通道号 <file path> 默认是 output 目录
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 out:	停止 A/D 数据分	

	ChronosInst 实例 error in: 错误输入	ChronosInst 实例 ret: 操作状态码 error out: 错误输出	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 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	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

	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	ChronosInst out: ChronosInst 实例 ret: 操作状态码	ES 数据分析并保存数据到文件 ● Process 分	<ch> 必须指定 A/D 通道号

	析持续时间 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 error in: 错误输入	error out: 错误输出	析数据文件	<file path> 默认是 output 目录
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	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

	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 error in: 错误输入			<file path> 默认是 output 目录
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	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

	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			<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
ciFetchConformESProcessData	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 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
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	ChronosInst out: ChronosInst 实例	实时获取 Conform ES Area2 Process 数	1）Process 数据是分析处理后的统计数据；

	error in: 错误输入	data: Int64 数值数组 error out: 错误输出	据（高通道号）	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	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 ● ch2 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 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

	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			
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） 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	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
ciFetchPeriodRAWData	ChronosInst in: ChronosInst 实例 timeout(ms)：超时时间	ChronosInst out: ChronosInst 实例 data： Int64 数值数组	实时获取 Period RAW 数据	1）在超时时间内，直到获取数据才返回；

	（单位：毫秒） error in: 错误输入	error out: 错误输出		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	<duration delay> 默认 2 秒 <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			
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
ciFetchStartStopProcessData	ChronosInst in: ChronosInst 实例 start chan: 开始通道号 stop chan: 终止通道号 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 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
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	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	<duration delay> 默认 2 秒 <file path> 默认是 output 目录

	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		析数据文件 ● Start-stop Average Process 分析数据文件	
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

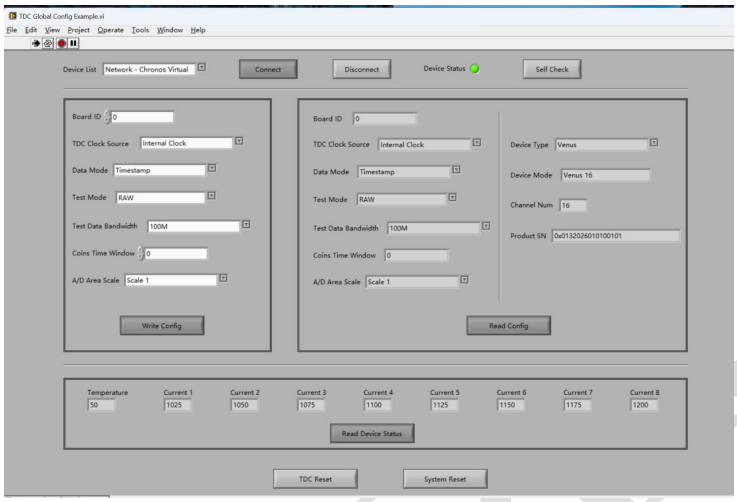
2) 线程安全：

- 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

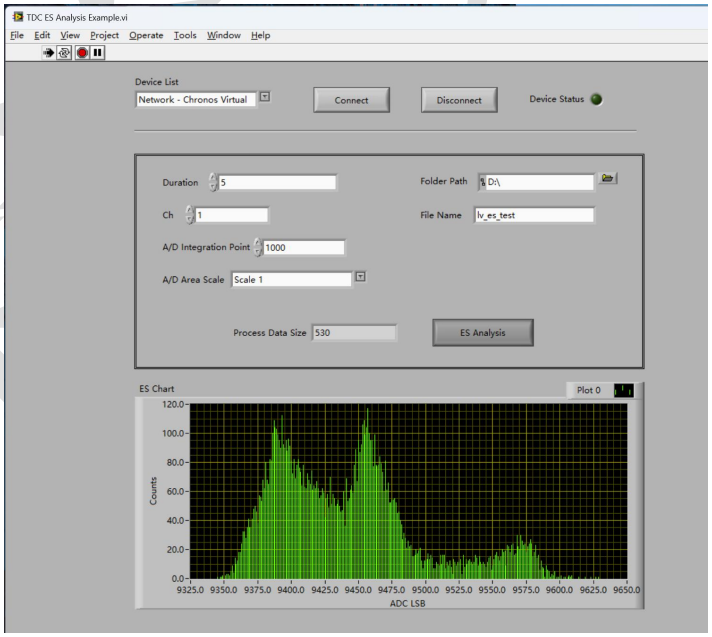
9.5.13 参考示例

请参考 Examples 目录下的示例。

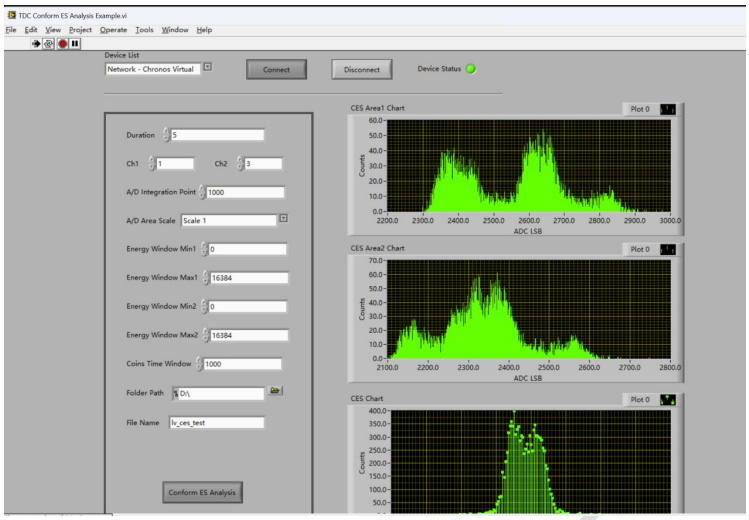
1) 全局配置 Example



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 8 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	单位
时间符合数据	global_coins.txt	通道号 1-通道号 2-时间差	ps
脉宽测量数据	tot.txt	通道号-脉宽值	ps
ADC 测量数据	adc.txt	通道号-AD 值	LSB
能谱测量数据	area.txt	通道号-能量	LSB

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	LSB-LSB-ps
双边沿时间戳数据	Dual_singles.txt	通道号-边沿类型-时间	ps

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”；

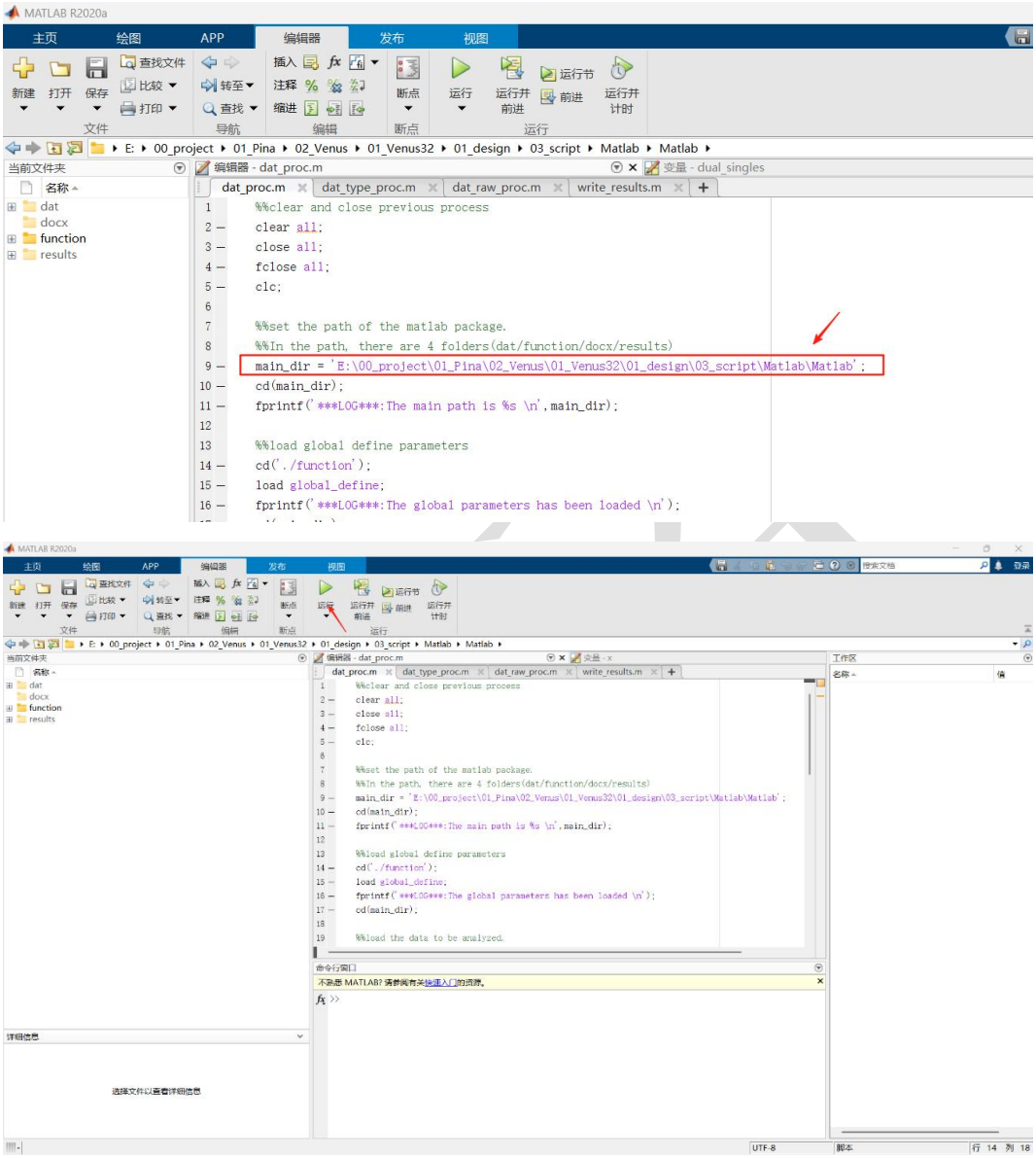


图 62 数据分析 MATLAB 脚本运行

（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”；

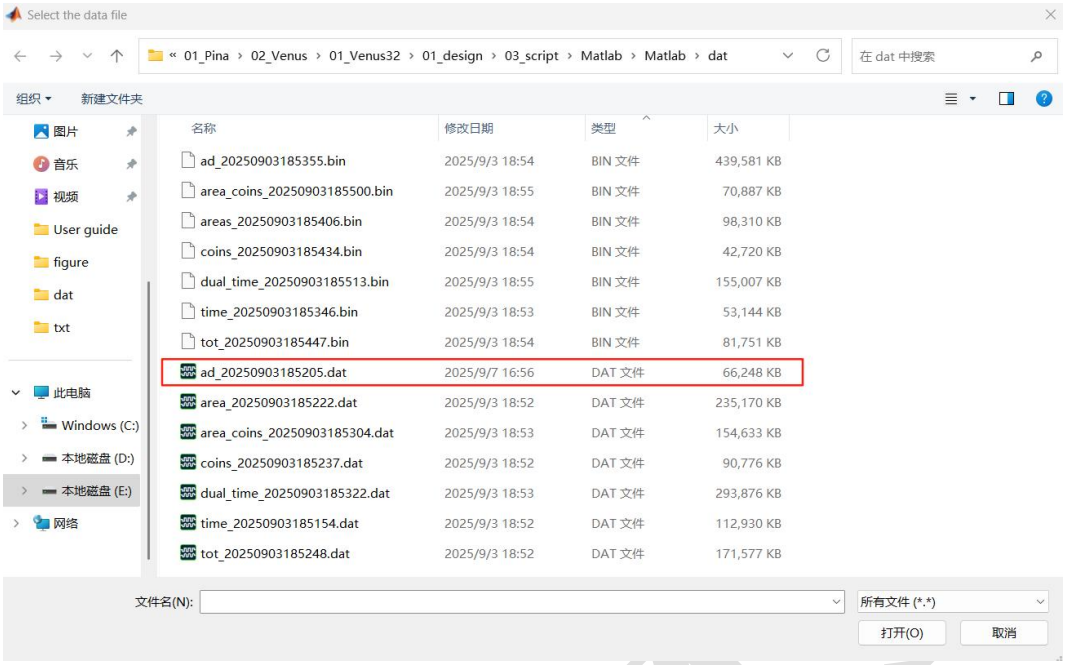


图 63 选择要分析的 adc Raw 数据文件

（3）程序会自动完成数据读取，格式转换和结果保存等；

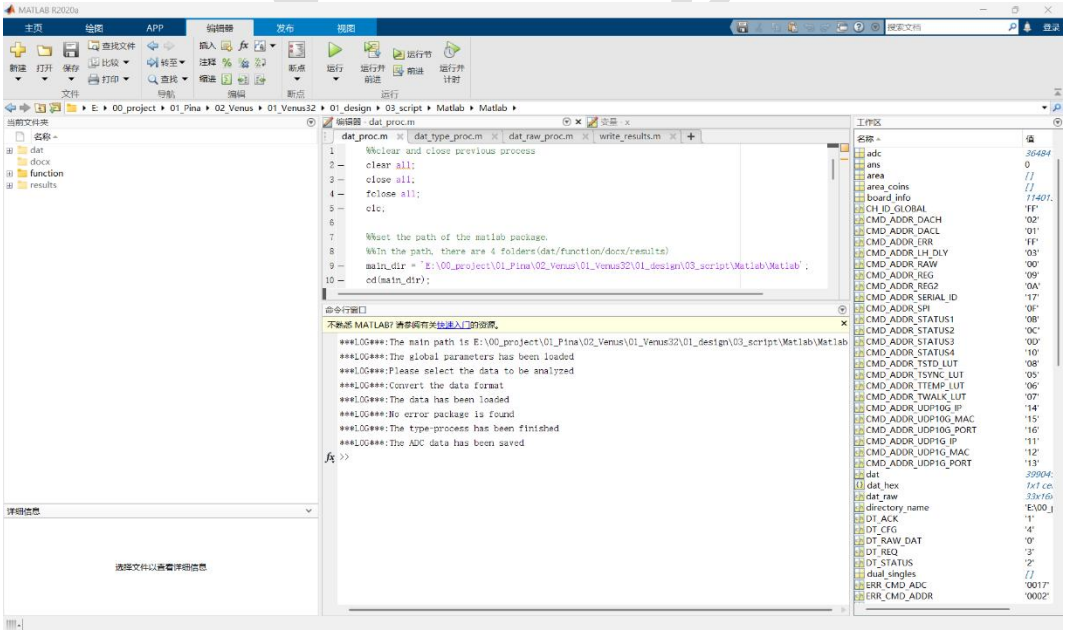


图 64 数据分析 MATLAB 脚本运行

（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。

图 65 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 ps。

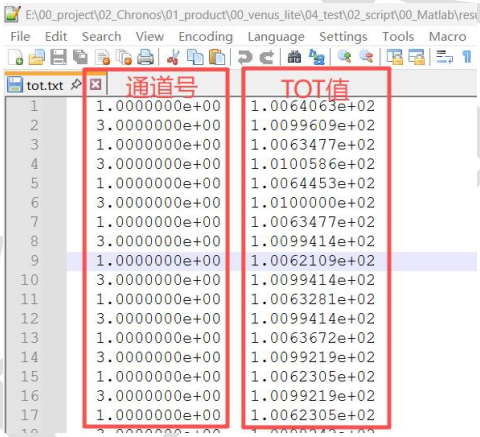
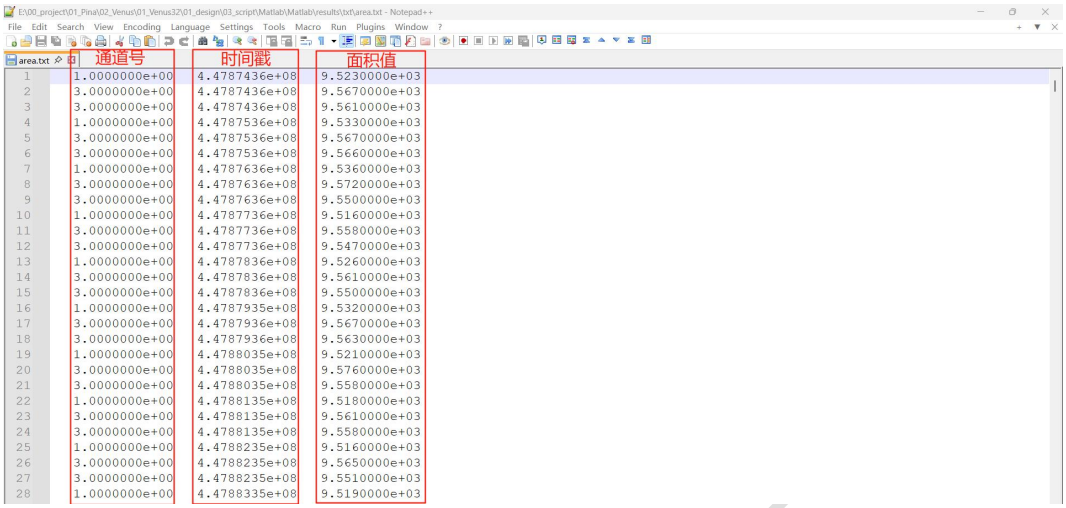


图 66 输出“tot.txt”数据格式

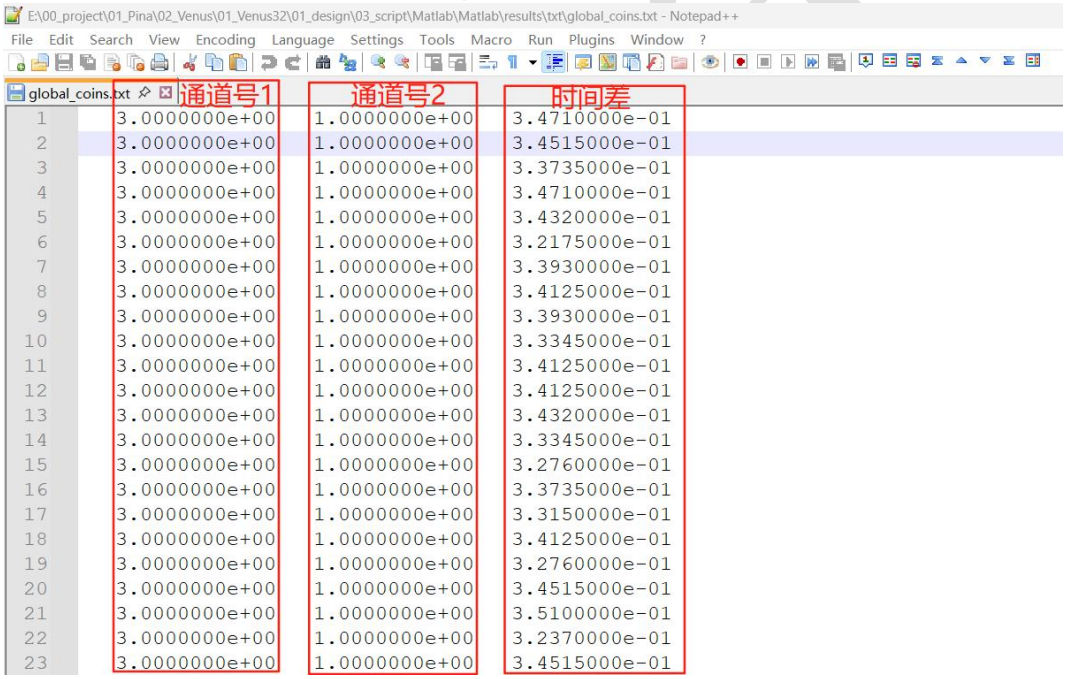
对于面积数据，第一列为通道号，第二列为时间戳计数值，单位为 ps，第三列为 area 值，单位为 ADC LSB。



	通道号	时间戳	面积值
1	1.0000000e+00	4.4787436e+08	9.5230000e+03
2	3.0000000e+00	4.4787436e+08	9.5670000e+03
3	3.0000000e+00	4.4787436e+08	9.5610000e+03
4	1.0000000e+00	4.4787536e+08	9.5330000e+03
5	3.0000000e+00	4.4787536e+08	9.5670000e+03
6	3.0000000e+00	4.4787536e+08	9.5660000e+03
7	1.0000000e+00	4.4787636e+08	9.5360000e+03
8	3.0000000e+00	4.4787636e+08	9.5720000e+03
9	3.0000000e+00	4.4787636e+08	9.5500000e+03
10	1.0000000e+00	4.4787736e+08	9.5160000e+03
11	3.0000000e+00	4.4787736e+08	9.5580000e+03
12	3.0000000e+00	4.4787736e+08	9.5470000e+03
13	1.0000000e+00	4.4787836e+08	9.5260000e+03
14	3.0000000e+00	4.4787836e+08	9.5610000e+03
15	3.0000000e+00	4.4787836e+08	9.5500000e+03
16	1.0000000e+00	4.4787935e+08	9.5320000e+03
17	3.0000000e+00	4.4787936e+08	9.5670000e+03
18	3.0000000e+00	4.4787936e+08	9.5630000e+03
19	1.0000000e+00	4.4788035e+08	9.5210000e+03
20	3.0000000e+00	4.4788035e+08	9.5760000e+03
21	3.0000000e+00	4.4788035e+08	9.5580000e+03
22	1.0000000e+00	4.4788135e+08	9.5180000e+03
23	3.0000000e+00	4.4788135e+08	9.5610000e+03
24	3.0000000e+00	4.4788135e+08	9.5580000e+03
25	1.0000000e+00	4.4788235e+08	9.5160000e+03
26	3.0000000e+00	4.4788235e+08	9.5650000e+03
27	3.0000000e+00	4.4788235e+08	9.5510000e+03
28	1.0000000e+00	4.4788335e+08	9.5190000e+03

图 67 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 ps。



	通道号1	通道号2	时间差
1	3.0000000e+00	1.0000000e+00	3.4710000e-01
2	3.0000000e+00	1.0000000e+00	3.4515000e-01
3	3.0000000e+00	1.0000000e+00	3.3735000e-01
4	3.0000000e+00	1.0000000e+00	3.4710000e-01
5	3.0000000e+00	1.0000000e+00	3.4320000e-01
6	3.0000000e+00	1.0000000e+00	3.2175000e-01
7	3.0000000e+00	1.0000000e+00	3.3930000e-01
8	3.0000000e+00	1.0000000e+00	3.4125000e-01
9	3.0000000e+00	1.0000000e+00	3.3930000e-01
10	3.0000000e+00	1.0000000e+00	3.3345000e-01
11	3.0000000e+00	1.0000000e+00	3.4125000e-01
12	3.0000000e+00	1.0000000e+00	3.4125000e-01
13	3.0000000e+00	1.0000000e+00	3.4320000e-01
14	3.0000000e+00	1.0000000e+00	3.3345000e-01
15	3.0000000e+00	1.0000000e+00	3.2760000e-01
16	3.0000000e+00	1.0000000e+00	3.3735000e-01
17	3.0000000e+00	1.0000000e+00	3.3150000e-01
18	3.0000000e+00	1.0000000e+00	3.4125000e-01
19	3.0000000e+00	1.0000000e+00	3.2760000e-01
20	3.0000000e+00	1.0000000e+00	3.4515000e-01
21	3.0000000e+00	1.0000000e+00	3.5100000e-01
22	3.0000000e+00	1.0000000e+00	3.2370000e-01
23	3.0000000e+00	1.0000000e+00	3.4515000e-01

图 68 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 ps。

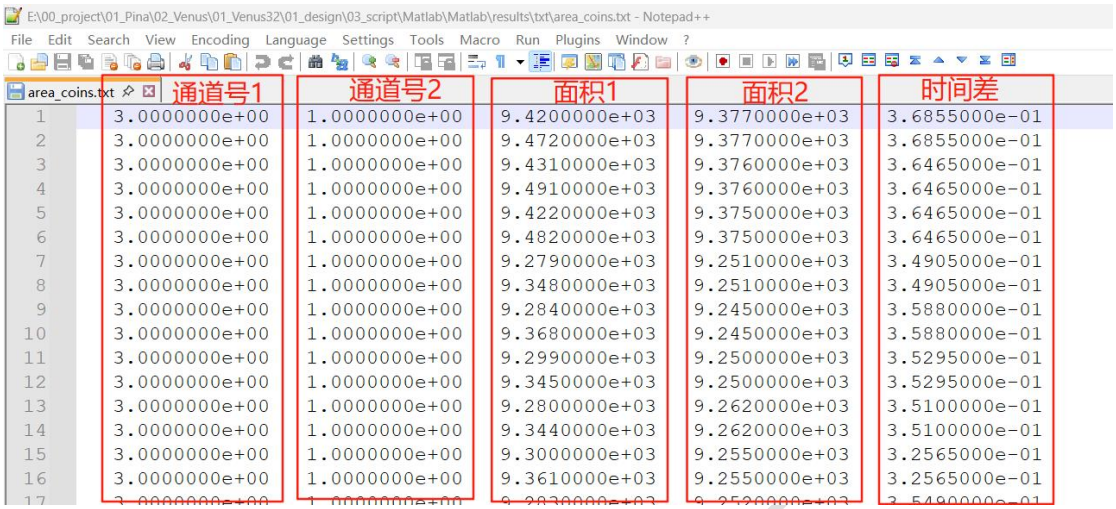


图 69 输出“area_coins.txt”数据格式

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 ps。

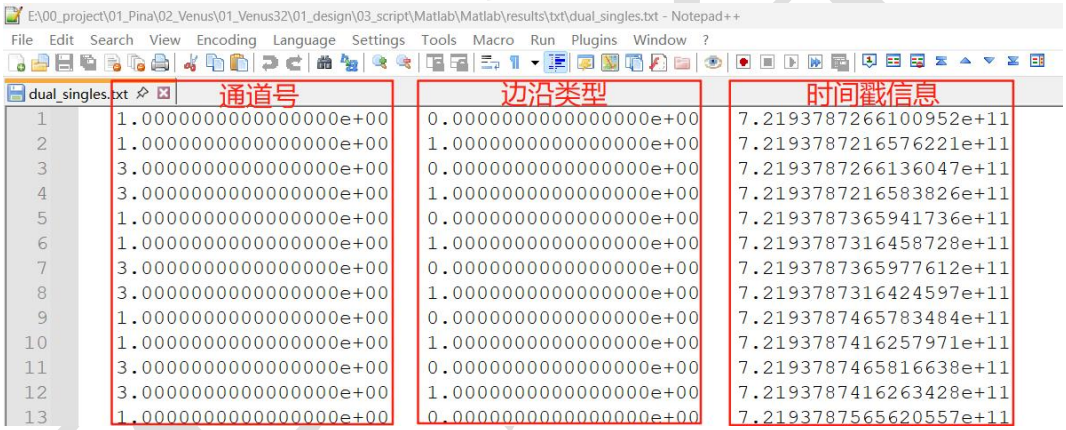


图 70 输出“dual_singles.txt”数据格式

对于周期测量数据，第一列为通道号，第二列为类型（1：高电平时间，0：下周期时间），第三列为时间戳计数值，单位为 ps。

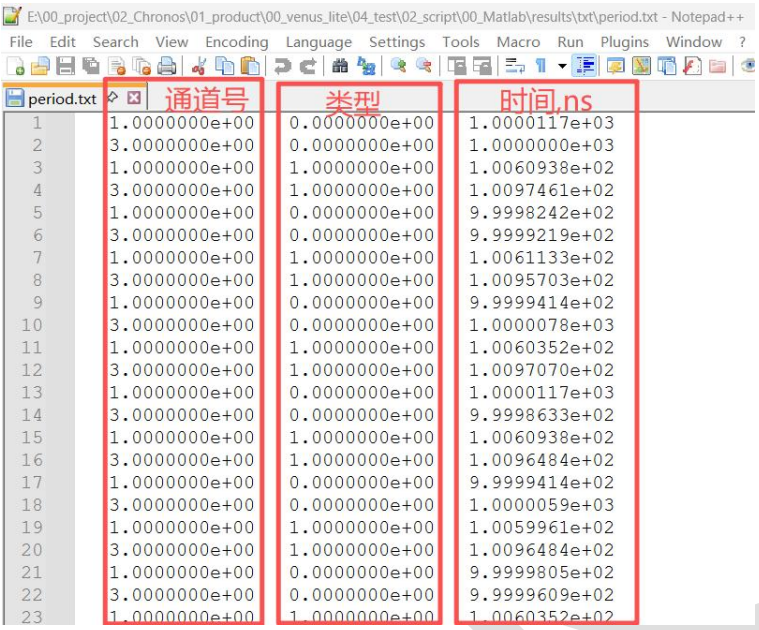


图 71 输出“period.txt”数据格式

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 9 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
		main.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。

```
================================================================================
主工作路径： E:\00_project\02_Chronos\01_product\00_venus_lite\04_test\02_script\02_PYTHON
***LOG***: 全局参数已成功加载
***LOG***: 请通过文件选择对话框指定待分析的数据文件
***LOG***: 选择的文件： E:/00_project/02_Chronos/01_product/00_venus_lite/04_test/02_script/02_PYTHON/dat/global_coins_20
251207130254.bin
***LOG***: 正在读取.bin二进制文件...
***LOG***: 数据加载完成。数据行数： 2187220
***LOG***: 数据加载与格式转换已完成
***LOG***: 正在解析数据类型...
数据总行数： 2187220
找到帧头数量： 62492
找到帧尾数量： 62492
找到有效raw数据帧数量： 62492
dat_raw形状： (32, 16, 62492)
***LOG***: 数据类型解析与分类已完成
***LOG***: 正在分析原始数据...
=== dat_raw_proc 开始处理 ===
输入数据形状： (32, 16, 62492)
帧数： 62492
数据区域总行数： 1937252
总共找到双单数据： 0 条

=== 处理结果统计 ===
板卡信息： 62492 条
全局符合： 1937252 条
参考符合： 0 条
ADC数据： 0 条
面积数据： 0 条
TOT数据： 0 条
面积符合： 0 条
双单数据： 0 条
周期数据： 0 条
***LOG***: 原始数据分析完成
  - 板卡信息条目： 62492
  - 全局符合事件： 1937252
  - 参考符合事件： 0
  - ADC数据条目： 0
  - 面积数据条目： 0
  - TOT数据条目： 0
  - 面积符合数据： 0
  - 双单数据： 0
  - 周期数据： 0
***LOG***: 正在保存结果...
***LOG***: The global coins data has been saved
***LOG***: 所有结果保存完成
```

图 72 Raw 数据分析 python 脚本分析过程

10. 典型测试电路和测试结果

本章主要介绍 Venus 系统的测试方案与核心结果。本章将以 Venus 16lite-T 型号为例进行详细说明。

注意，Venus lite 系列（不带-T）为非自然场景（无对流或者极低噪音要求）下使用，用户应将设备放置于大面积的导热平面上，以保证温度的散热良好。Venus lite-T 无此使用要求。

10.1 系统标定

为准确设置触发阈值，建议在首次使用设备或外界环境（如温度）发生显著变化时，先执行一次系统标定，再设置具体的阈值。

本次标定中，设备输入端口处于悬空状态。标定参数设置如下：

- 扫描范围： -100 mV ~ +100 mV
- 步进值： 1 mV
- 平均次数： 5

标定结果如下图所示。需要注意的是，增加平均次数可有效降低随机噪声的影响，从而提高标定结果的准确度和稳定性，但是需要更多的标定时间。

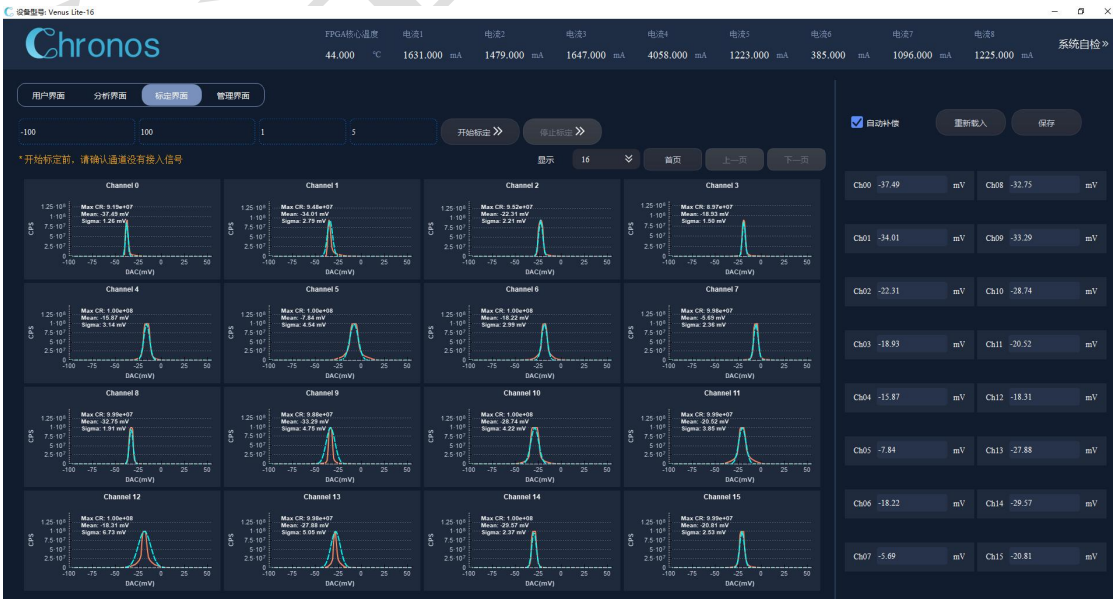


图 73 系统标定结果（标定范围-100 mV 到+100 mV，步进值 1 mV，平均次数 5，典型结果）

点击“自动补偿”后，在用户界面将各通道阈值统一设置为目标值（分别为 10 mV 和 50 mV）。 注意：用户需要重新执行阈值配置后，自动补偿才会生效。标定补偿前后的阈值一致性测试结果对比如下图所示。

可以看出，经过系统标定与自动补偿，各通道阈值的一致性得到了显著改善。但由于 DAC 的非线性、量化误差、拟合误差，以及外界环境干扰和测试系统误差等因素，阈值仍存在一定的离散性，此点需要用户特别注意。建议阈值设置在 50 mV 以上。

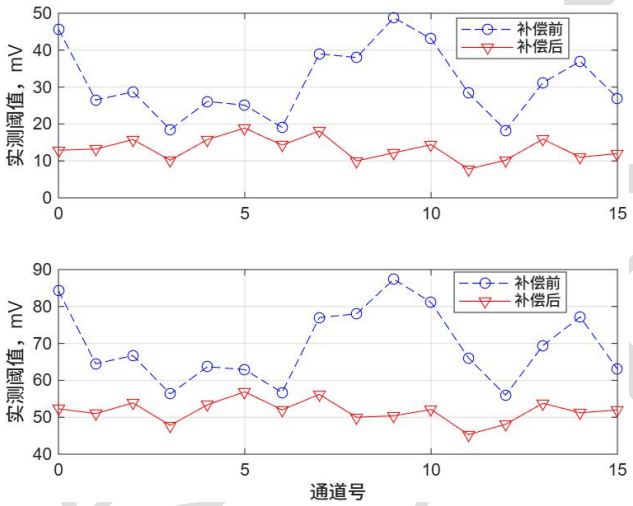


图 74 实测阈值电压结果（补偿前与补偿后，目标阈值分别为 10 mV 和 50 mV）

10.2 双通道时间差测量（内部触发）

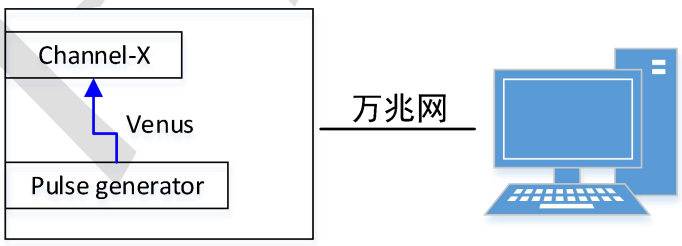


图 75 双通道时间差测量示意图（内部触发）

在用户界面中启用测试模式并选择内部触发后，Venus 会由一个独立于 TDC 时钟的电路产生一个周期性测试信号（重复频率：2 MHz/通道）。各时间测量通道会对该信号进行测量并输出时间值。触发边沿为上升沿。

测量数据经过在线符合处理后，通过万兆以太网接口上传至上位机。用户可

在软件的分析页面中，使用双通道时间差测量功能进行在线分析，也可采集数据进行离线分析，以获取通道间的时间差分布。

图 76 高分辨率通道双通道时间差分布直方图典型测试结果（内部触发，上升沿，CH01）

图 77 普通通道双通道时间差分布直方图典型测试结果（内部触发，上升沿，CH45）

本次测量环境温度为 25 °C。

- 高精度测量通道（Venus Lite 系列为 CH0~CH3）：本次对 CH0 与 CH1 的测量结果表明，其符合时间差抖动的标准差（σ） 典型值约为 4.3 ps (RMS)。由此推算出的单通道定时精度为 $\sigma/\sqrt{2} \approx 3.0$ ps (RMS)。
- 普通测量通道：本次对 CH4 与 CH5 的测量结果如下图所示，其符合

时间差抖动的标准差（σ） 约为 7.2 ps (RMS)，对应的单通道定时精度为 σ/√2 ≈ 5.1 ps (RMS)。

注意：由于硬件参数的离散性、TDC 的固有量化误差以及信号在 FPGA 内部走线延迟不同，不同通道间的时间晃动测量结果会存在差异。

下表为本设备典型状态下的时间分辨率测试结果（注：仅供参考，具体性能以实际测量为准）：

- 高精度通道平均定时精度：~3 ps (RMS)
- 普通通道平均定时精度：~6 ps (RMS)

表格 10 某机器典型时间分辨率测试结果（内部触发，典型结果）

通道选择	CH0/1	CH2/3	CH4/5	CH6/7	CH8/9	CH10/11	CH12/13	CH14/15
上升沿 σ/√2	3.0 ps	3.1 ps	5.6 ps	5.6 ps	5.5 ps	5.3 ps	6.1 ps	4.4 ps
下降沿 σ/√2	3.2 ps	3.1 ps	5.8 ps	6.2 ps	5.3 ps	5.5 ps	6.2 ps	5.1 ps
正中间沿 σ/√2	2.6 ps	2.5 ps	4.2 ps	4.2 ps	4.6 ps	4.5 ps	5.0 ps	4.2 ps
负中间沿 σ/√2	2.5 ps	2.2 ps	4.3 ps	4.5 ps	4.1 ps	4.8 ps	5.1 ps	4.3 ps

10.3 双通道时间差测量（外部触发）

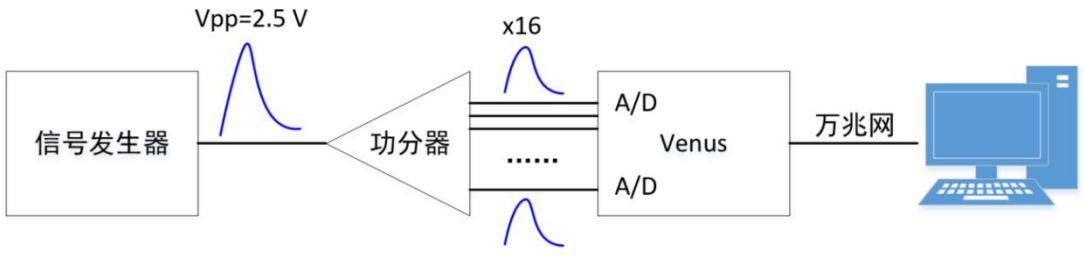


图 78 双通道时间差测量示意图

一、测试配置与参数

- 信号源： 信号发生器输出正脉冲，具体参数如下：
  - 重复频率： 1 MHz
  - 上升时间： 1 ns
  - 脉冲宽度： 5.8 ns
  - 幅度： 2.5 V
- 信号分配： 输出信号经由一个射频无源功分器分成 16 路，分别接入 Venus 系统的两个测试通道。理论上，到达每个通道的信号幅度约为 600 mV。
- 系统设置： 将这两个输入通道的时间比较阈值均设置为 200 mV，触发边沿类型为上升沿。

二、数据处理与分析

测量数据经系统在线符合处理后，通过万兆以太网接口上传至上位机。用户可通过上位机软件执行以下分析：

- 在线分析： 使用分析页面的双通道时间差测量功能进行实时分析。
- 离线分析： 采集并保存全局符合数据，进行更深入的离线处理，最终得到两通道间的时间差分布谱。

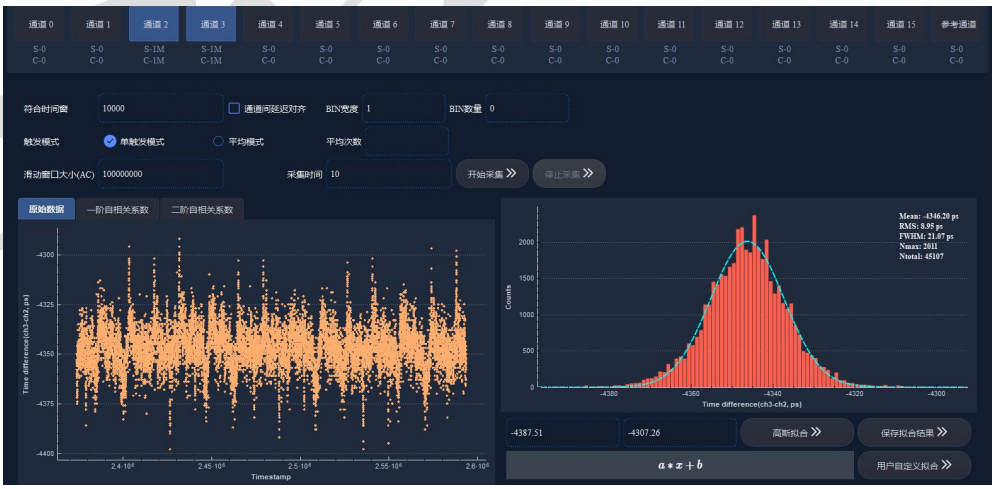


图 79 双通道时间差外部输入信号测量结果（高分辨率通道，通道 2 和 3，上升沿，典型结果）

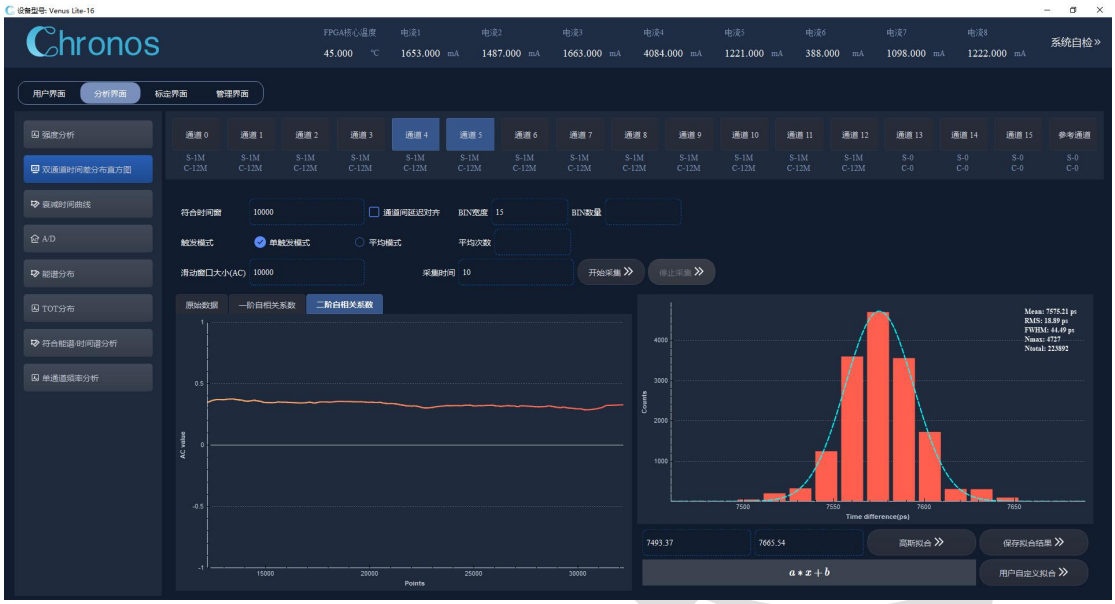


图 80 双通道时间差外部输入信号测量结果（普通通道，通道 4 和 5，上升沿，典型结果）

本次测量环境温度为 25 °C。

- 高精度测量通道（Venus Lite 系列为 CH0~CH3，本次测量通道为 CH0 与 CH1）：时间差分布如下图所示，符合时间差抖动的标准差（σ）约为 9.7 ps (RMS)，推算的单通道定时精度为 σ/√2 ≈ 6.9 ps (RMS)；
- 普通测量通道（本次测量通道为 CH4 与 CH5）：时间差分布如下图所示，符合时间差抖动的标准差（σ）约为 11.5 ps (RMS)，推算的单通道定时精度为 σ/√2 ≈ 8.2 ps (RMS)。

重要提示：

上述时间晃动测量结果与外部输入信号特性密切相关。当输入信号幅度越大且上升沿越陡峭（即过阈值点的斜率越大，或称摆率越高）时，时间测量精度越高。关于输入信号摆率、噪声与测量精度之间关系的详细讨论，请参阅附录 E。

时间延迟补偿只是为了将通道间的延迟对齐，方便在多通道应用中设置统一的符合时间窗，并不会改善时间晃动性能。在一般应用中，用户可参考以下步骤进行整个系统的通道间的延迟标定和对齐。

首先，将符合时间窗开尽量大，让所有通道间的符合数据都能收集；

然后，计算/拟合各个通道与某一个通道间的时间差平均值；

然后，补偿每个通道的延迟值；

最终，重新测试数据，验证对齐的正确性。

注意：由于硬件参数的固有离散性及 TDC 的量化误差，不同通道间的时间晃动测量结果存在差异。

下表提供了典型时间分辨率测试结果（注：仅供参考，具体性能以实际测量为准）：

- 高精度通道平均定时精度：~ 7 ps (RMS)
- 普通通道平均定时精度：~ 11 ps (RMS)

表格 11 某机器典型时间分辨率测试结果（外部触发）

通道选择	CH0/1	CH2/3	CH4/5	CH6/7	CH8/9	CH10/11	CH12/13	CH14/15
上升沿 σ/√2	6.9 ps	7.2 ps	13.3 ps	9.7 ps	9.2 ps	11.3 ps	10.5 ps	11.5 ps

10.4 单通道重复频率测量

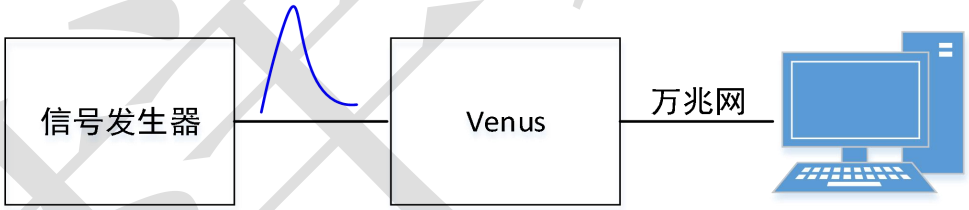


图 81 单通道周期信号重复频率测量示意图

使用信号发生器，进行单通道重复频率测量。信号发生器设置输出信号为正脉冲，重复频率 5 MHz，信号上升沿时间为 3 ns，下降沿时间为 3 ns，占空比 50%，幅度 2 V。输入到 Venus 系统一个测量通道中。Venus 系统这两路通道的时间比较阈值设置为 200 mV，然后经过在线处理后，通过万兆网接口将数据上传到上位机中。注意，频率分析功能与选择边沿触发类型无关。

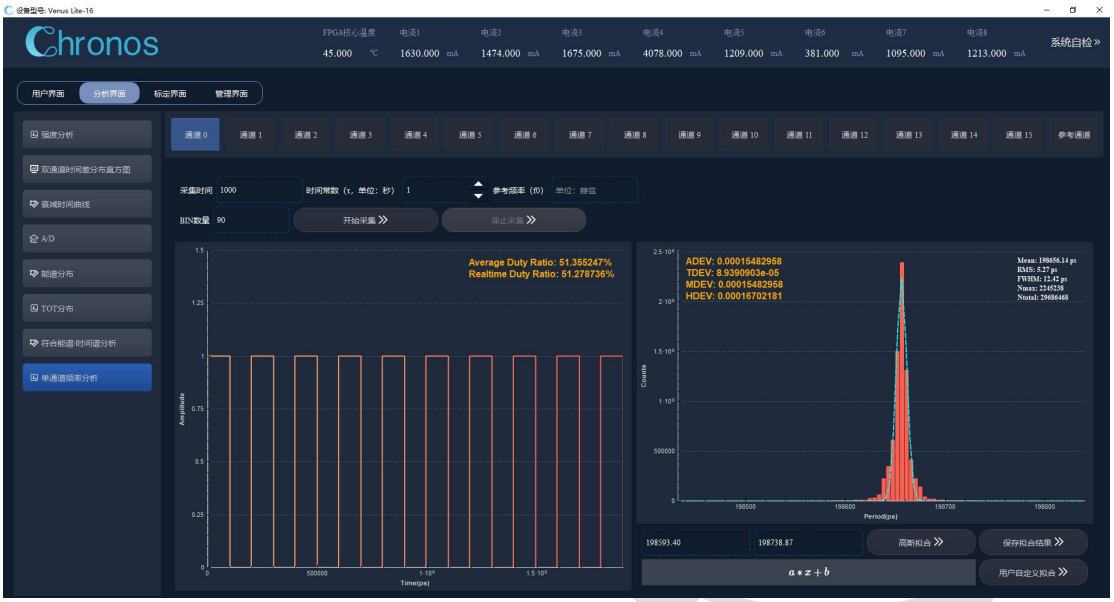


图 82 单通道重复频率测量

上位机软件计算信号周期、占空比、周期抖动等并显示。

10.5 单通道脉宽测量

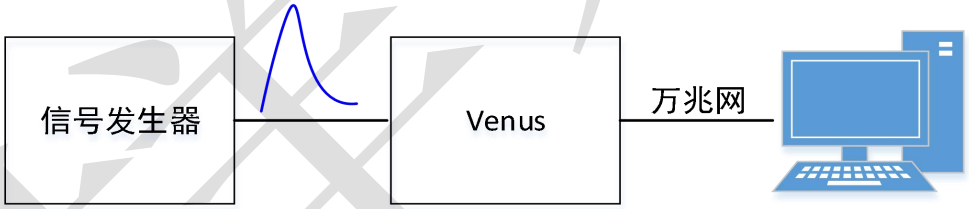


图 83 单通道信号脉宽测量示意图

一、测试配置与参数

信号源： 信号发生器输出正脉冲，具体参数如下：

- 重复频率： 1 MHz
- 上升/下降时间： 3 ns
- 脉冲宽度： 100 ns (标称值)
- 幅度： 2 V
- 系统连接与设置： 将该信号输入至 Venus 的一个测量通道，并将该通道的时间比较阈值设置为 1000 mV，边沿触发类型选择上升沿

二、数据处理与分析

测量数据经系统在线处理后，通过万兆以太网接口上传至上位机。通过分析软件，可计算出脉冲宽度的平均值及其晃动值（标准偏差）。

三、测试结果

测试条件： 环境温度 25 °C。

- 高精度测量通道 (CH0)：脉宽测量平均值： 100.7 ns，脉宽晃动值: 12.9 ps (RMS)；
- 普通测量通道 (CH4)：脉宽测量平均值： 100.1 ns，脉宽晃动值: 17.2 ps (RMS)

图 84 TOT 外部输入信号测量结果（高分辨率通道，通道 0，典型结果）

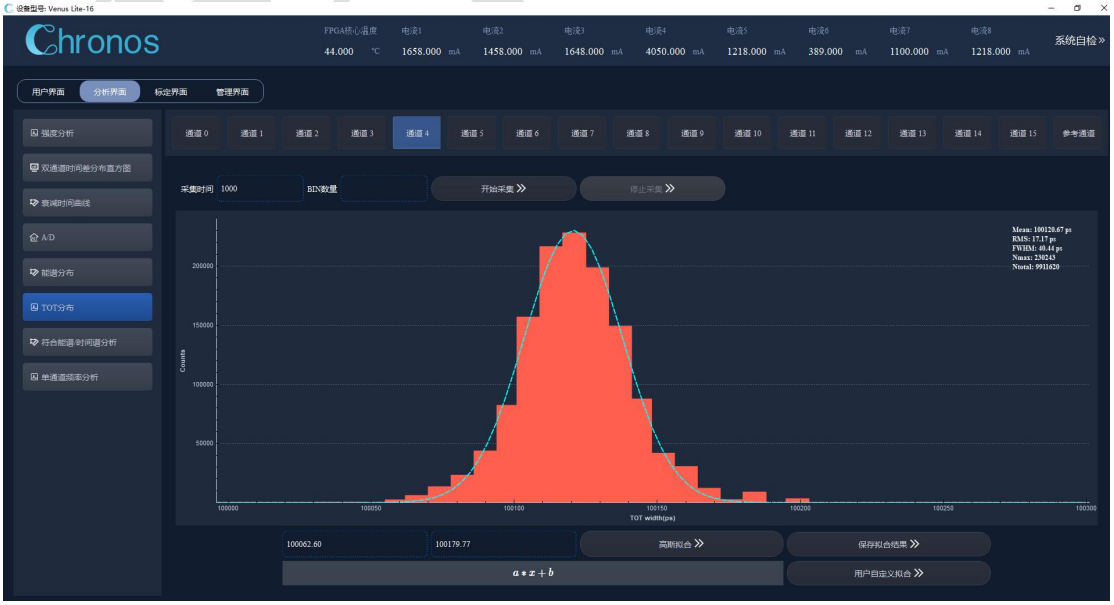


图 85 TOT 外部输入信号测量结果（普通通道，通道 4，典型结果）

注意：由于输入信号本身的抖动、硬件参数的离散性及 TDC 的固有量化误差，各通道的 TOT（信号脉宽）测量结果会存在差异。

下表为本设备典型状态下的 TOT 测量晃动测试结果（注：仅供参考，具体性能以实际测量为准）：

- 高精度通道平均 TOT 晃动：~ 12 ps (RMS)
- 普通通道平均 TOT 晃动：~18 ps (RMS)

重要提示：上述结果为系统总晃动，其中包含了输入信号自身的抖动。因此，Venus 系统本身引入的测量晃动小于该测试值。

表格 12 TOT 外部输入信号典型测量结果

通 道 选择	CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7
σ	12.9 ps	13.5 ps	9.5 ps	12 ps	17.2 ps	21.8 ps	18.4 ps	15.8 ps
通 道 选择	CH8	CH9	CH10	CH11	CH12	CH13	CH14	CH15
σ	15.3 ps	14.6 ps	21.8 ps	19.7 ps	17 ps	15 ps	17.6 ps	17.8 ps

10.6 模拟-数字转换实时波形测量

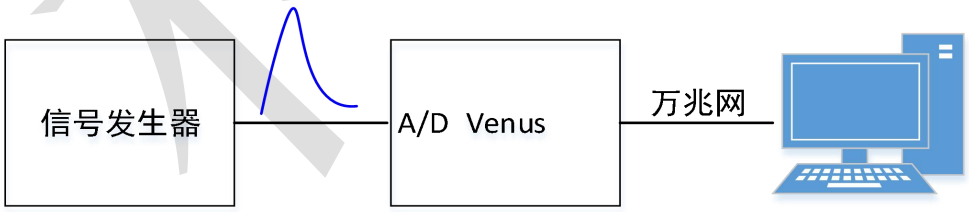


图 86 模拟-数字转换实时波形测量

本测试使用信号发生器验证 Venus 系统的模拟信号采集与波形显示功能。

一、测试配置与参数

信号源： 信号发生器输出正弦波，具体参数如下：

- 频率： 5 MHz
- 幅度： -2 V ~ +2 V (峰峰值 4 V)

系统连接与设置：

- 将信号输入至 Venus 的一个 ADC 测量通道（通道 1）；
- 将该通道的时间比较阈值设置为 0 mV（注：在此功能中，阈值仅用于触发，波形由 ADC 记录）；
- 设置 ADC 积分点数（记录长度） 为 1000。

二、数据处理与分析

测量数据经系统在线处理后，通过万兆以太网接口上传至上位机。用户可通过上位机软件执行以下分析：

- 在线观测： 使用分析页面中的 “模拟-数字转换实时波形测量” 功能观测实时信号波形；
- 离线分析： 保存 ADC 原始数据，进行更灵活的离线处理与波形重现。

本次测量环境温度为 25 °C。

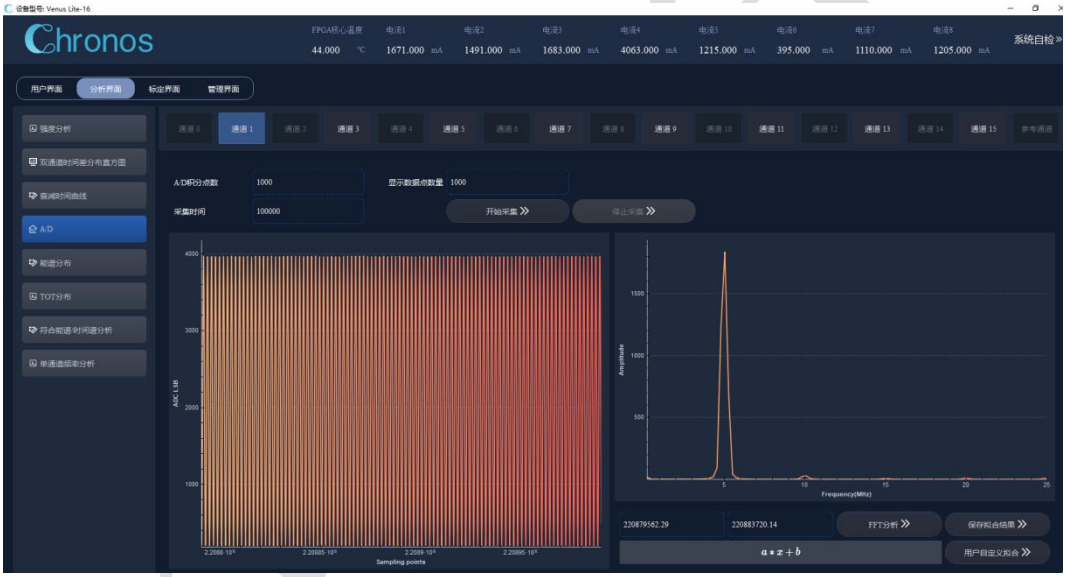


图 87 模拟-数字转换实时波形测量（输入正弦波，频率 5 MHz，幅度-2V - +2V，通道 1，典型结果）

本测试使用信号发生器验证 Venus 系统对脉冲波形的采集与显示功能。

一、测试配置与参数

信号源： 信号发生器输出正脉冲，具体参数如下：

- 重复频率： 1 MHz
- 上升/下降时间： 3 ns

- 脉冲宽度： 100 ns
- 幅度： +2 V
- 触发边沿选择上升沿
- 将信号输入至 Venus 的一个 ADC 测量通道（通道 1）
- 将该通道的时间比较阈值设置为 100 mV（用于触发）
- 设置 ADC 积分点数（记录长度） 为 30。

二、数据处理与分析

测量数据经系统在线处理后，通过万兆以太网接口上传至上位机。用户可通过上位机软件执行以下分析：

- 在线观测： 使用 “模拟-数字转换实时波形测量” 功能观测实时信号波形。
- 离线分析： 保存 ADC 原始数据进行深入的波形分析。

环境温度为 25 °C。

采集到的脉冲波形如下图所示。

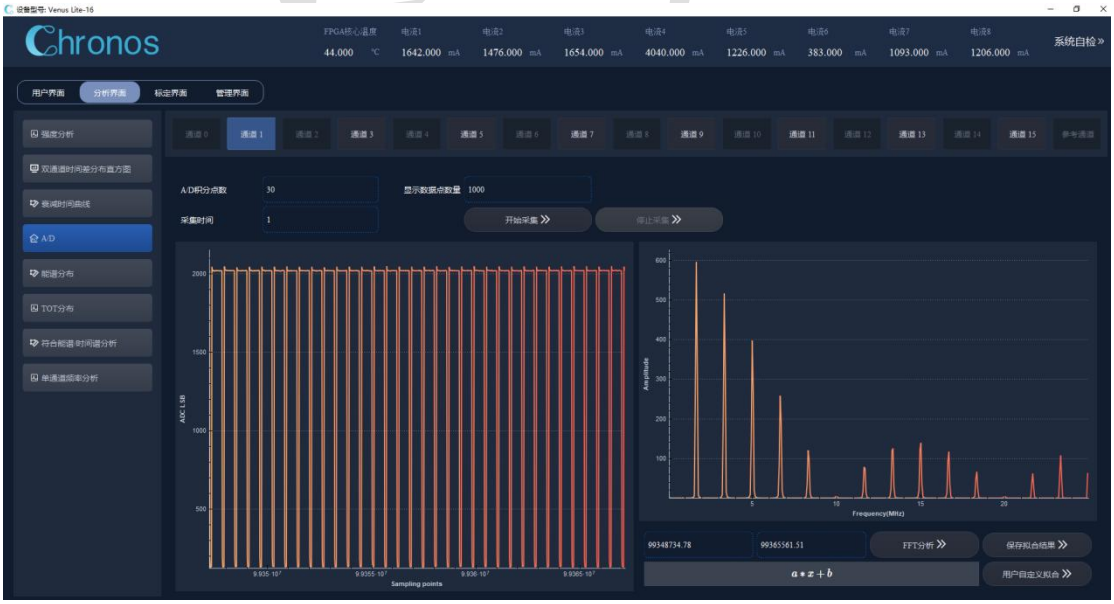


图 88 模拟-数字转换实时波形测量（输入正脉冲，频率 1 MHz，幅度+2V，通道 1，典型结果）

此外，Venus 系统同样支持对负脉冲波形的采集与测量。

一、测试配置与参数

信号源： 信号发生器输出负脉冲，具体参数如下：

- 重复频率： 1 MHz
- 上升/下降时间： 3 ns
- 脉冲宽度： 100 ns
- 幅度： -2 V
- 触发边沿为下降沿
- 将信号输入至 Venus 的一个 ADC 测量通道（通道 1）；
- 将该通道的时间比较阈值设置为 -100 mV（用于负脉冲触发）；
- 设置 ADC 积分点数（记录长度） 为 30。

二、数据处理与分析

测量数据经系统在线处理后，通过万兆以太网接口上传至上位机。用户可通过上位机软件执行以下分析：

- 在线观测： 使用 “模拟-数字转换实时波形测量” 功能观测实时信号波形；
- 离线分析： 保存 ADC 原始数据进行深入的波形分析。

测试条件： 环境温度为 25 °C。

采集到的负脉冲波形如下图所示。

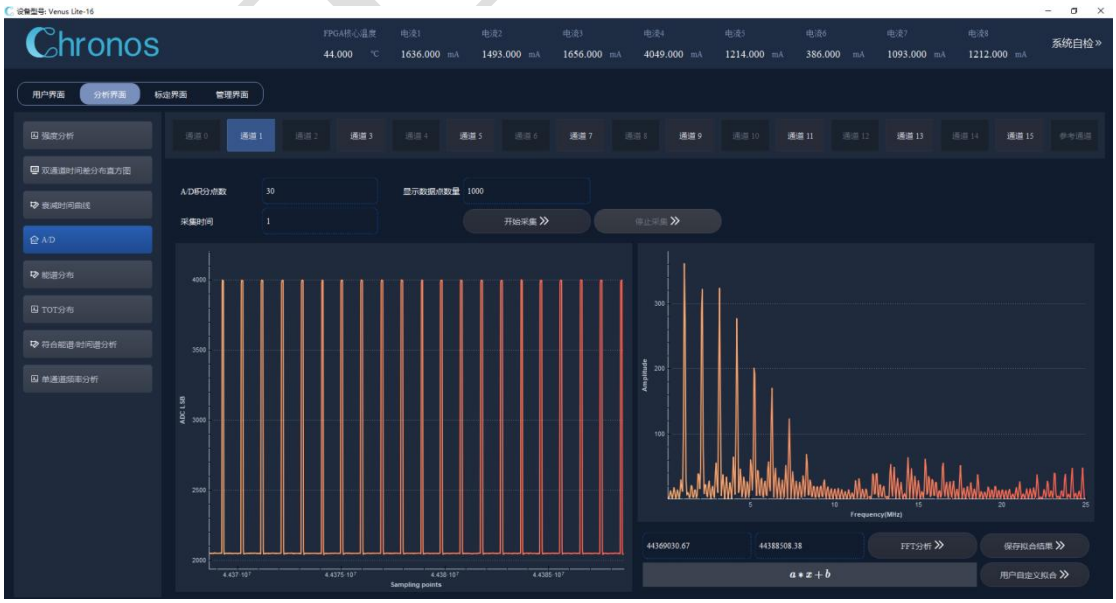


图 89 模拟-数字转换实时波形测量（输入负脉冲，频率 1 MHz，幅度-2 V，通道 1，典型结果）

10.7 能谱测量

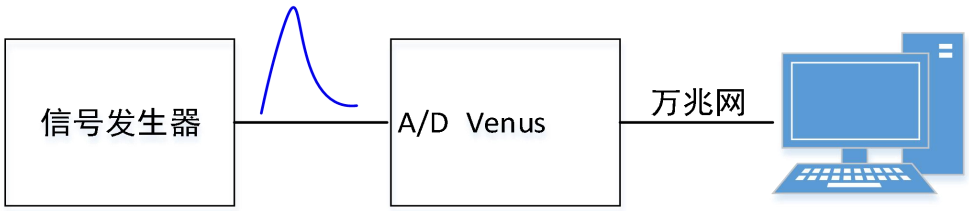


图 90 能谱测量示意图

一、测试配置与参数

信号源： 信号发生器输出正脉冲，具体参数如下：

- 重复频率： 1 MHz
- 上升/下降时间： 3 ns
- 脉冲宽度： 100 ns
- 幅度： 2 V

系统连接与设置：

- 将信号输入至 Venus 的一个 ADC 测量通道；
- 将该通道的时间比较阈值设置为 100 mV（用于触发）；
- 边沿触发类型选择上升沿
- 设置 ADC 积分点数（积分门宽） 为 30。

二、数据处理与分析

测量数据经系统在线处理后，通过万兆以太网接口上传至上位机。用户可通过上位机软件执行以下分析：

- 在线观测： 使用分析页面中的“能谱测量”功能观测实时能谱分布；
- 离线分析： 保存 能谱原始数据进行深入的谱分析。

环境温度为 25 °C。

采集到的能谱分布如下图所示。

此外，软件支持面积值缩放功能，可对每个脉冲的积分面积（能量值）进行 1 至 128 倍的缩放。针对高能量信号或设置较长积分门宽时，脉冲积分面积可能非常大。启用缩放功能可有效防止能量值在数据传输与处理过程中发生溢出

（“饱和”），确保测量数据的完整性与准确性。

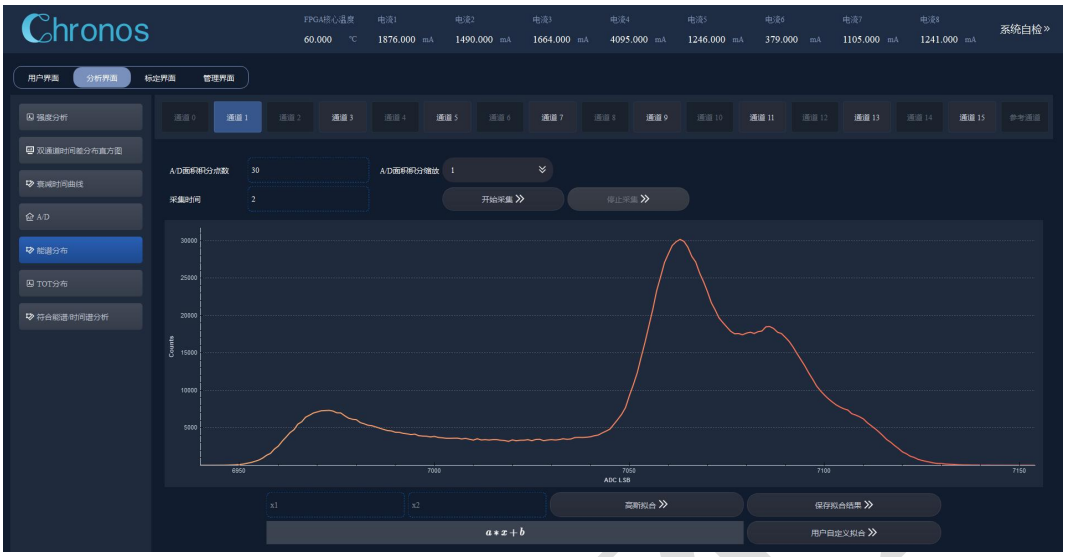


图 91 能谱外部输入信号测量结果（输入正脉冲，频率 1 MHz，幅度+2V，通道 1，典型结果）

此外，Venus 系统同样支持对负脉冲信号的能谱测量。

一、测试配置与参数

信号源： 信号发生器输出负脉冲，具体参数如下：

- 重复频率： 1 MHz
- 上升/下降时间： 3 ns
- 脉冲宽度： 100 ns
- 幅度： -2 V

系统连接与设置：

- 将信号输入至 Venus 的一个 ADC 测量通道；
- 将该通道的时间比较阈值设置为 -100 mV（用于负脉冲触发）；
- 边沿触发类型选择上升沿
- 设置 ADC 积分点数（积分门宽） 为 30。

二、数据处理与输出

测量数据经系统在线处理（包括脉冲积分与能谱生成）后，通过万兆以太网接口上传至上位机，供用户进行在线或离线能谱分析。

图 92 能谱外部输入信号测量结果（输入负脉冲，频率 1 MHz，幅度-2V，通道 1，典型结果）

10.8 全局时间-能谱符合测量

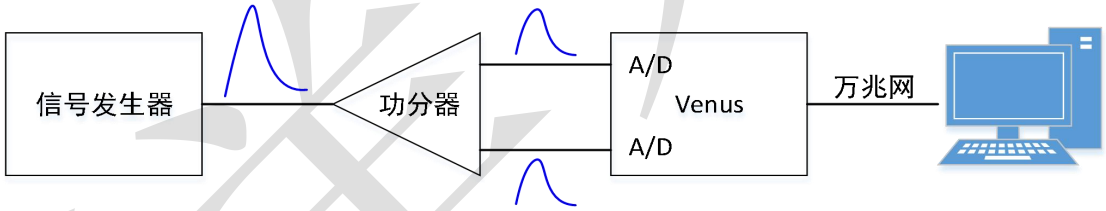


图 93 全局时间-能谱符合测量示意图

本测试采用信号发生器与线延迟法进行双通道时间差与能谱性能测量。

一、测试配置与参数

信号源： 信号发生器输出正脉冲，具体参数如下：

- 重复频率： 1 MHz
- 上升/下降时间： 3 ns
- 脉冲宽度： 100 ns
- 幅度： 4 V

信号分配与连接： 输出信号经由一个射频无源功分器分成两路，分别接入 Venus 系统的两个测试通道。

系统设置： 将这两个输入通道的时间比较阈值均设置为 1000 mV，触发边

沿类型为上升沿。

二、数据处理与分析

测量数据经系统在线符合处理后，通过万兆以太网接口上传至上位机。用户可通过上位机软件执行以下分析：

- 在线分析： 使用分析页面中的 “时间-能谱全局符合” 功能进行实时分析；
- 离线分析： 采集并保存全局符合数据，进行更深入的离线处理。

通道 1 与通道 3 的能谱分布如下图所示。

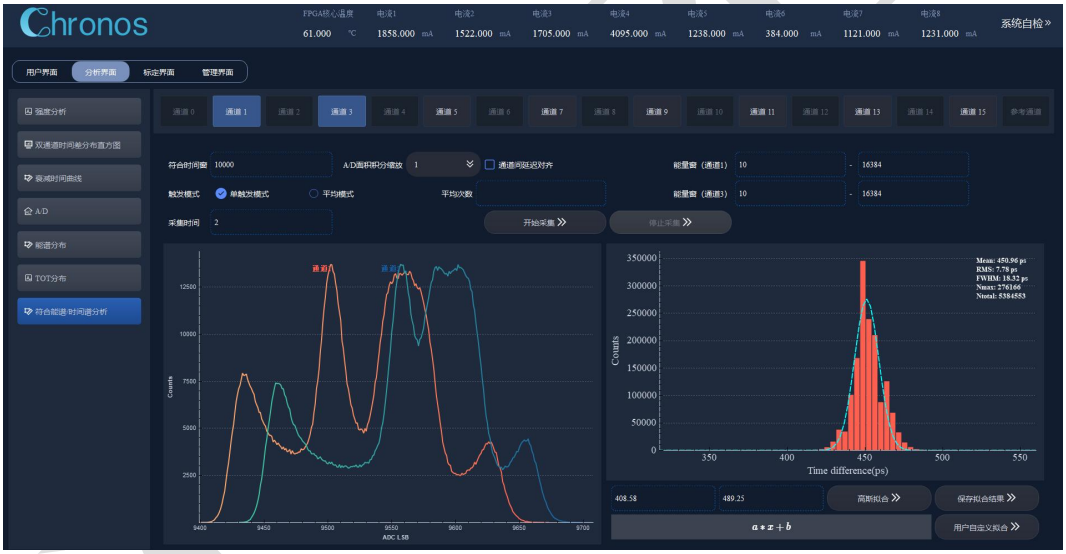


图 94 全局时间-能谱符合测量结果（输入正脉冲，频率 1 MHz，幅度+4V，通道 1 和 3，典型结果）

10.9 死时间测试

Venus 设备的死时间特性大致遵循非瘫痪（非扩展）死时间模型。可通过测量输入-输出计数率（ICR-OCR）曲线来验证各通道的死时间参数。该测试可在软件的“强度分析” 页面中完成。

一、测试配置与参数

测试原理： 通过扫描输入信号频率，测量系统在不同死时间设置下的输出

计数率（OCR）。

信号源： 信号发生器输出正脉冲，具体参数如下：

- 重复频率： 从低到高扫描，最高至 20 MHz
- 上升/下降时间： 3 ns
- 脉冲宽度： 100 ns
- 幅度： 2 V

系统连接与设置：

- 将信号输入至 Venus 系统的通道 0；
- 将通道 0 的目标死时间设置为 8 ns 至 500 ns（注：通常为设置一个值进行扫描，此处描述似为设置范围）；
- 将该通道的时间比较阈值设置为 1000 mV，触发边沿类型为上升沿。

二、数据处理与输出

测量数据经系统在线处理后，通过万兆以太网接口上传至上位机。

系统输出计数率（OCR）与死时间设置的关系如下表所示。实测结果与非瘫痪死时间模型的理论预期相符。

表格 13 不同死时间与 OCR 关系测试结果（ICR 为 20 Mcps，通道 0）

目标死时间，ns	8	20	52	100	200	500
OCR，Mcps	20	20	10.2	10	5	2

10.10 环境温度与 FPGA 核心温度关系

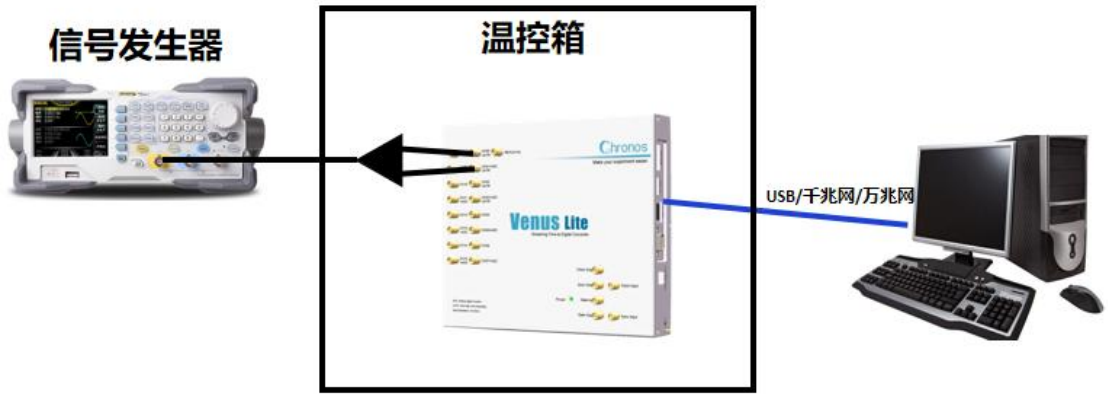


图 95 温度测试平台示意图

为测试 Venus 设备在不同环境温度下的性能稳定性，我们将其置于恒温箱中进行连续上电温漂测试。本测试使用带风扇版本 Venus lite-T。

测试方法如下：

- 设备布置： 将 Venus 设备放置于恒温箱内。
- 温度控制： 调整恒温箱，设定其目标温度。
- 温度监测： 待温度稳定后，通过上位机软件读取并记录设备 FPGA 的核心温度。
- 数据记录： 在不同目标温度点下重复上述步骤，以建立环境温度与 FPGA 核心温度的对应关系曲线。

温度测试使用的温度变化曲线（一个循环）如下所示。温度梯度为 0.333℃/min，在个别温度上保持时间为 30 分钟。每一台出厂设备均经过至少 3 次温度循环实验，以保证产品工作稳定性。

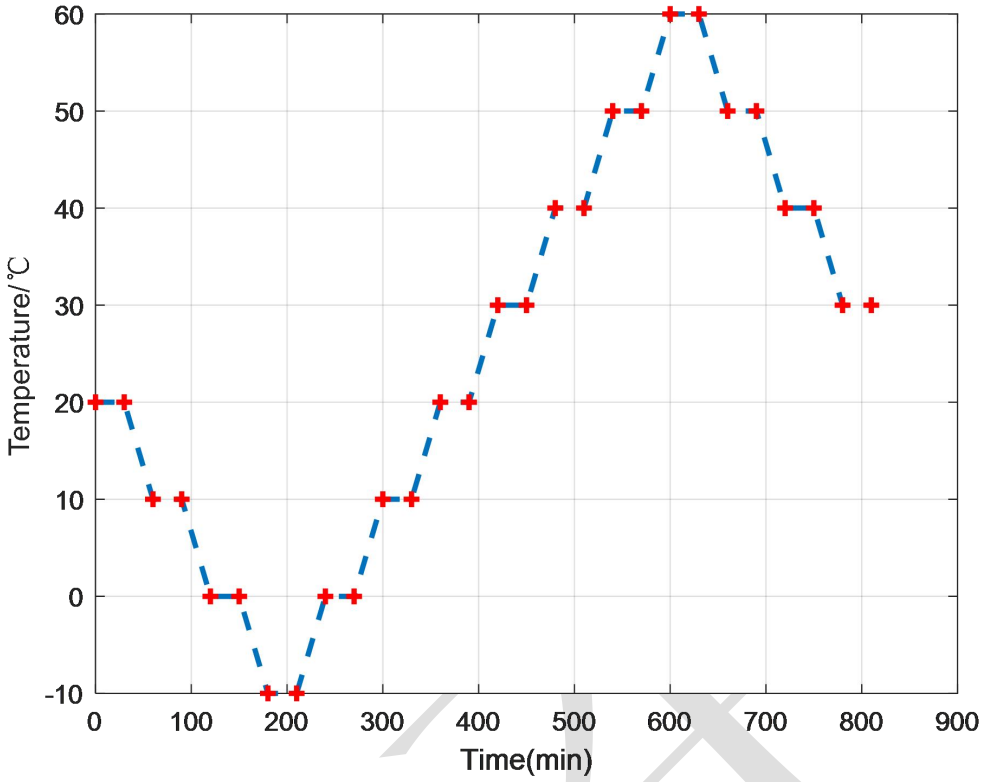


图 96 温度循环实验温度曲线设置

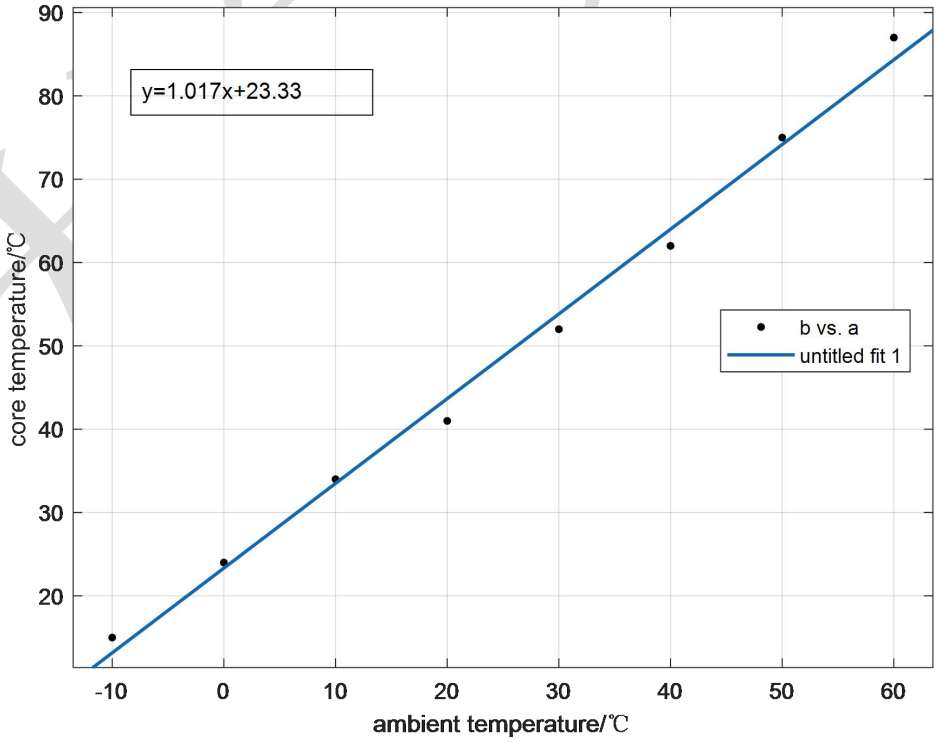


图 97 环境温度与 FPGA 内核稳定温度典型测试结果（Venus lite-T）

在各个温度下，设备多次上下电启动，均能够正常启动和建立连接。

10.11 时间晃动的温漂（内部触发）

为评估温度变化对 Venus 设备时间测量稳定性的影响，我们将其置于恒温箱中进行测试。

测试方法如下：

- 设备布置： 将 Venus 设备放置于恒温箱内;
- 系统配置： 将设备设置为内部触发模式（即使用其自产生的周期性测试信号）。

温度控制与测量：

- 调整恒温箱，设定一个目标温度;
- 待温度充分稳定后，进行数据采集;
- 数据分析： 通过上位机软件获取并分析双通道时间差分布，记录其中心值（峰位） 和分布宽度（标准差 σ 或 FWHM）;
- 迭代测试： 改变恒温箱的目标温度，重复步骤 3-4，以获取不同温度下的时间差分布参数，进而分析其温漂特性。

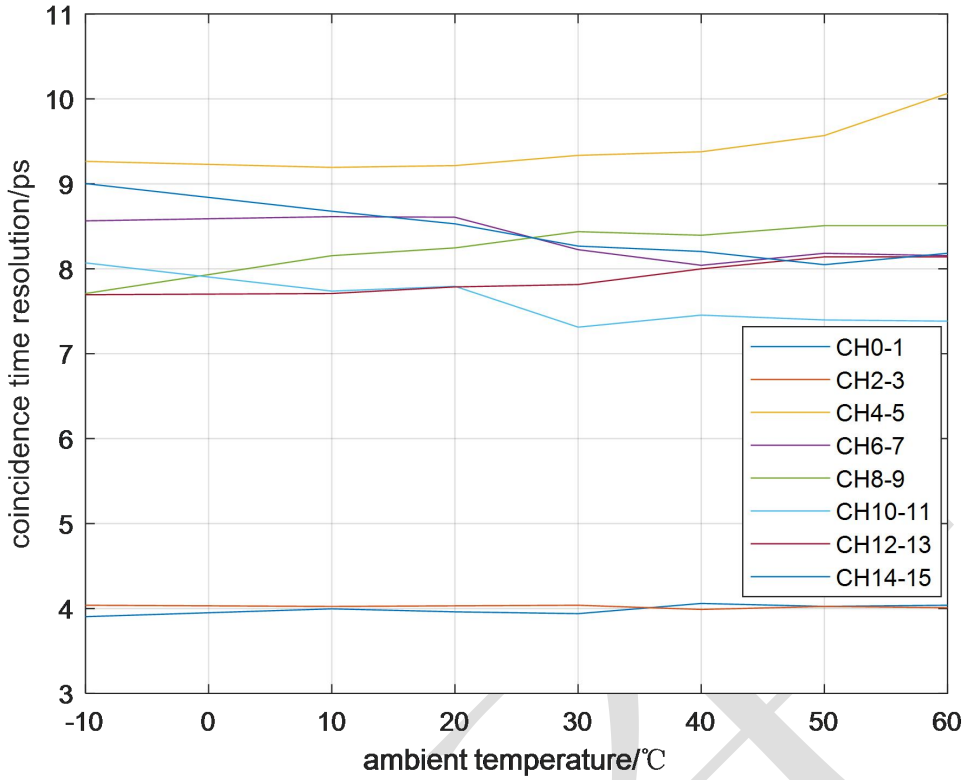


图 98 双通道时间差晃动与环境温度关系测试典型结果

测试结果表明，在-10 度 - 60 度环境温度下，设备的高分辨率通道时间晃动温漂值小于 ±0.002 ps/℃；普通通道时间晃动温漂值小于±0.02 ps/℃。

10.12 系统标定性能温漂

为评估 Venus 设备模拟前端的温度稳定性，我们将其置于恒温箱中，通过系统标定来测量其信号基线与阈值电压的温漂系数。调整恒温箱目标温度，设备进入系统标定模式，然后通过上位机软件读取标定结果，来判断信号基线和阈值电压温漂。

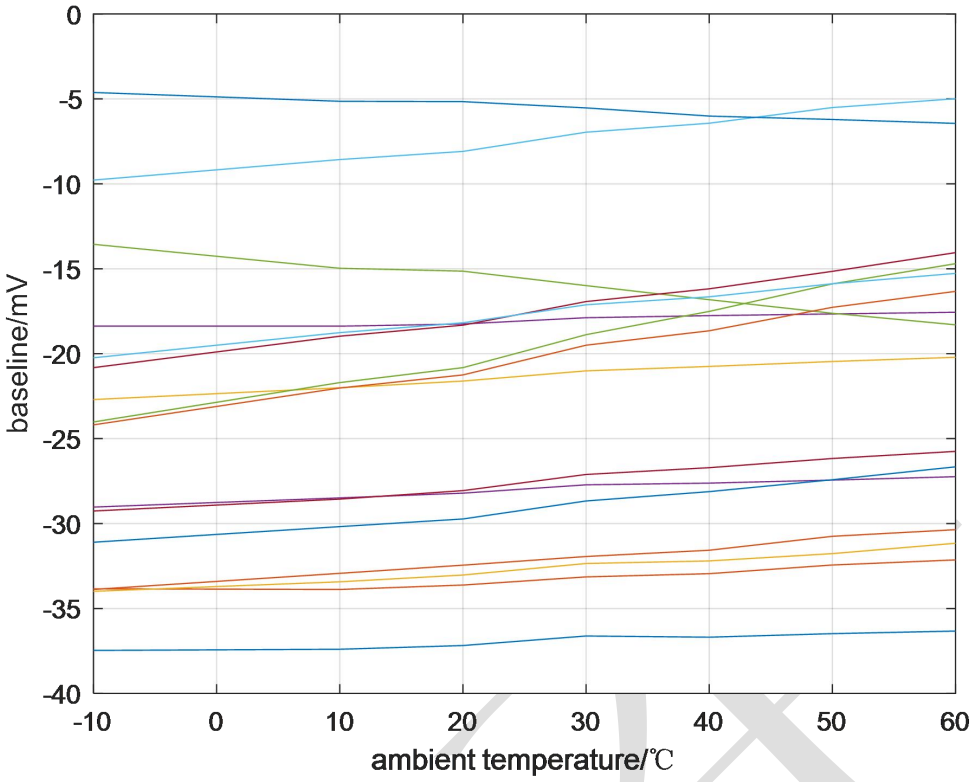


图 99 Venus 设备的系统标定基线平均值温漂典型测试结果

测试结果表明，系统标定的基线温漂小于±0.072 mV/℃。如此小的温漂，即使在-10°到＋60 度范围内，基线温漂也不超过±5 mV，对于一般应用来讲，用户不需要自己反复标定。

10.13 各通道与参考通道延迟温漂

由于各个测量通道包含一个高速比较器，其延迟存在温漂。并且信号基线温漂和阈值设置 DAC 存在温漂都会造成在不同温度下，各个测量通道的时间延迟发生变化。我们利用参考通道（不包含比较器）进行温漂的大致范围测试。对于不同形状的输入信号来讲，其影响是不同的。在我们的当前测试条件下，测试结果表明，各个测量通道的时间延迟温漂小于 1 ps/℃。具体不同应用场景下，用户可以与官方联系进行标定。

10.14 内外部时钟与时间晃动

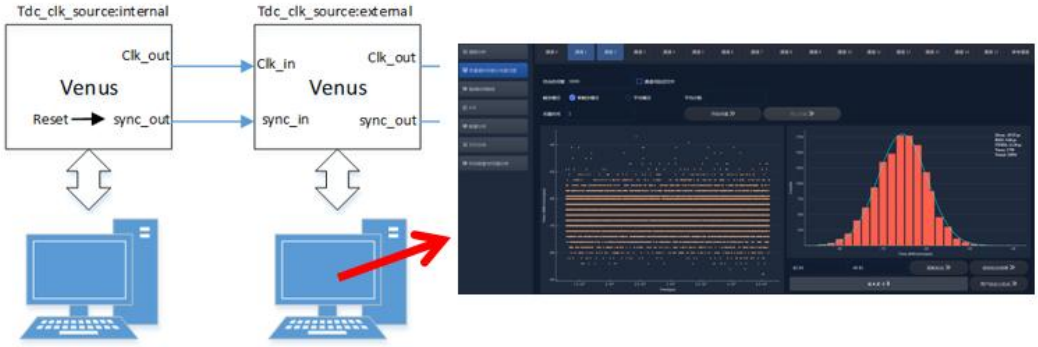


图 100 2 台 Venus 设备串联，第二台设备使用外部时钟进行时间晃动测试（内部触发）

使用如 11 章所述的，第一台 Venus 设备作为时钟源，驱动第二台 Venus 设备，如上图所示。第一台 Venus 设备 TDC 时钟配置为本地时钟，第二台 Venus 设备 TDC 时钟配置为外部时钟。第二台 Venus 设备设置为内部触发，测试第二台设备的 CH1 和 CH2 之间的时间差分布。测试表明，使用外部时钟，单通道时间晃动与使用内部时钟测试几乎相同。

图 101 Venus 设备 2 的 CH1 和 CH2 时间差分布（内部触发，上图：使用本地 TDC 时钟；下图：使用上一级 Venus 设备 1 的输出时钟）

10.15 时间晃动稳定性

在常温环境下，设备连续开机超过 24 h，相同设备参数配置下，每小时对外部触发模式进行双通道时间差晃动分析，来测试设备时间性能的稳定性。测试结果表明，在 24 小时内，各通道时间晃动差别不超过 0.5 ps。

10.16 能量测量线性度和精度

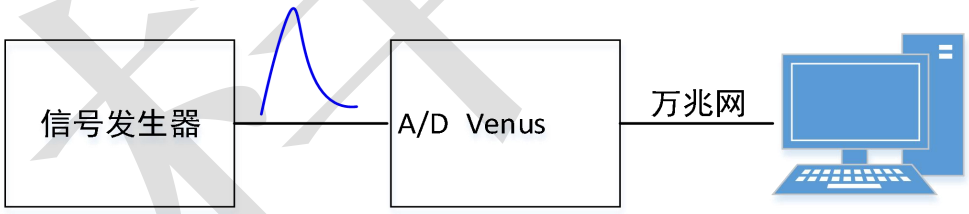


图 102 能谱测量示意图

本测试使用信号发生器验证 Venus 系统在不同幅度信号下的响应特性。

一、测试配置与参数

信号源： 信号发生器输出正脉冲，具体参数如下：

- 重复频率： 1 MHz
- 上升/下降时间： 3 ns
- 脉冲宽度： 100 ns
- 幅度： 30 mV - 2 V (扫描或变化)

系统连接与设置：

- 将信号输入至 Venus 的一个 ADC 测量通道（通道 1）
- 将该通道的时间比较阈值设置为 10 mV (用于触发)
- 边沿触发类型为上升沿
- 设置 ADC 积分点数（积分门宽） 为 20

二、数据处理与输出

测量数据经系统在线处理（包括脉冲积分与数据打包）后，通过万兆以太网接口上传至上位机，供后续分析使用。

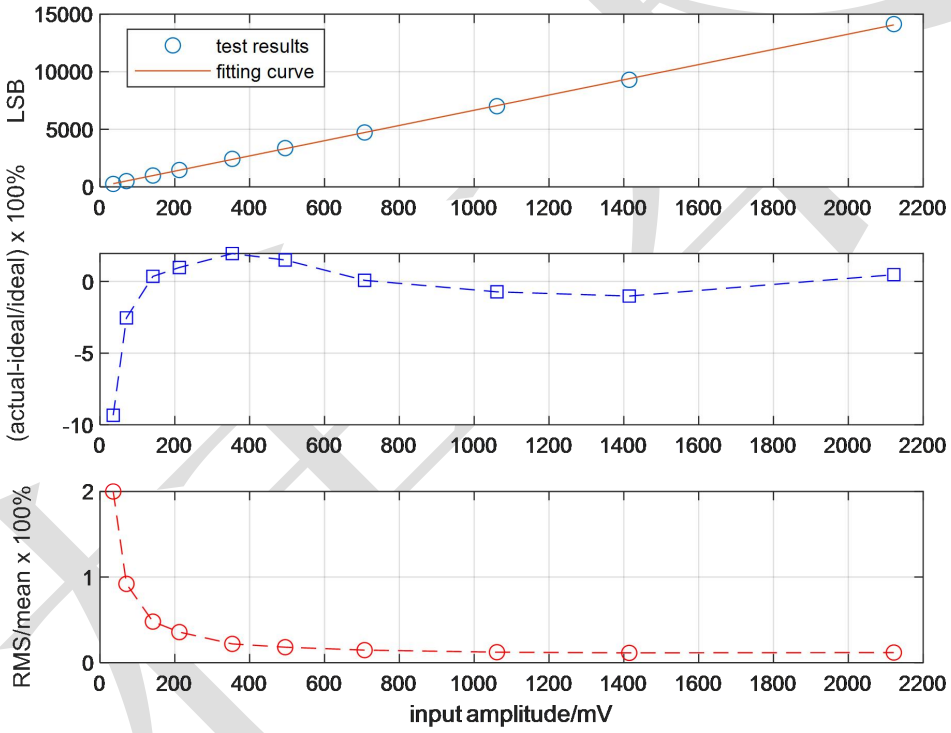


图 103 能量测量线性和分辨率测试典型结果（通道 1）

测试结果表明，针对本次测试的信号形状，在整个范围内，能量分辨率（RMS/mean, 100%）好于 2%。

11. 多设备时钟同步

11.1 多设备时钟同步

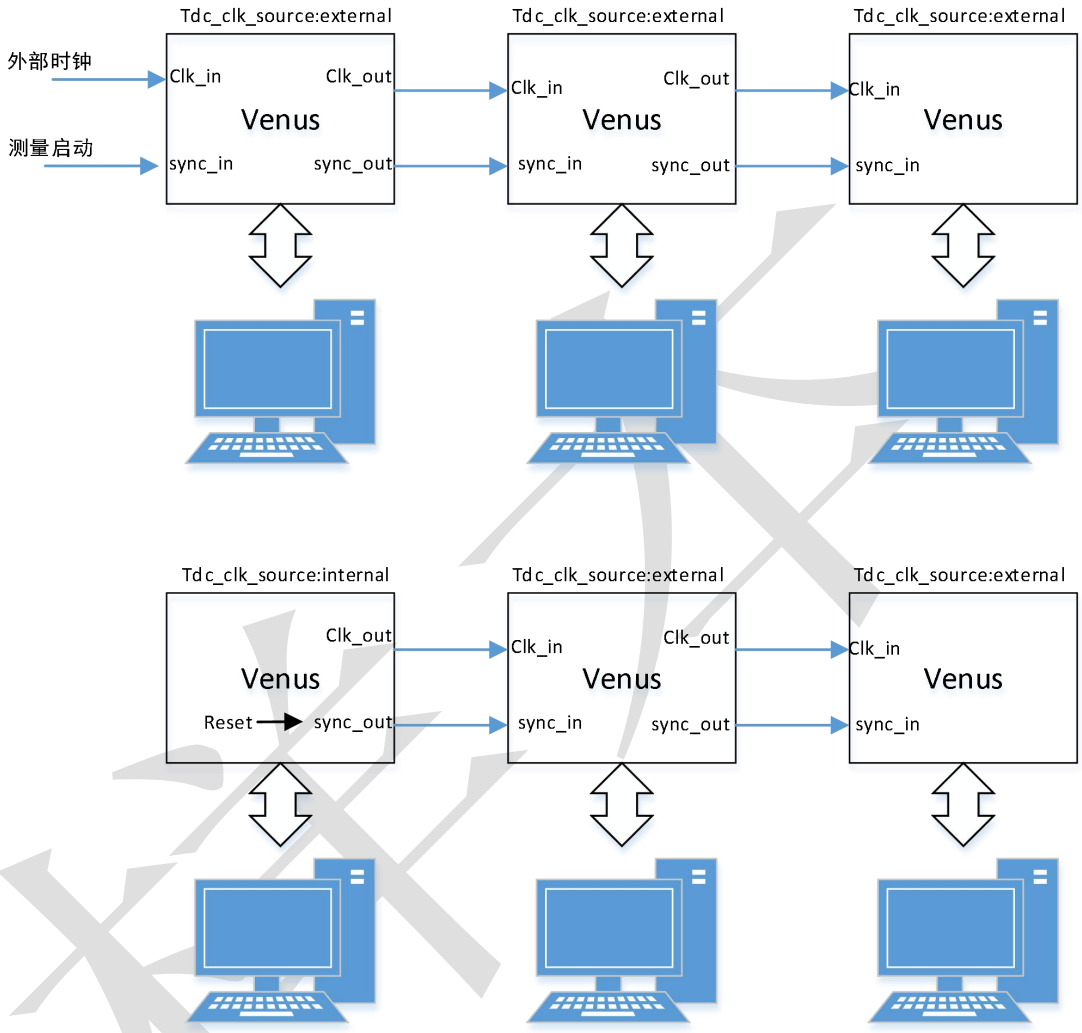


图 104 多设备时钟同步示意图（典型应用。上图：外部时钟，下图：第一个设备作为时钟源）

多个 Venus 设备可组建为一个高精度的同步测量网络。为实现系统内所有设备的时钟同步，各 Venus 设备均配备了以下接口：

- 外部时钟输入 (clk_in)
- 外部同步信号输入 (sync_in)
- 外部时钟输出 (clk_out)
- 外部同步信号输出 (sync_out)

一、同步配置方案

方案一：外接主时钟源同步

连接方式： 如上图所示。

系统配置：

- 所有 Venus 设备的 TDC 时钟源均设置为 “外部” 模式;
- 第一个 Venus 设备接收来自系统主时钟源的 10 MHz 时钟和同步信号;
- 第一个设备将时钟和同步信号分发给第二个设备，以此类推，形成串接链;
- 同步触发： 当 sync_in 接收到高电平脉冲（脉宽 ＞100 ns）时，链上的所有 TDC 将同时复位。

方案二：设备主时钟同步

连接方式： 如下图所示。

系统配置：

- 将第一个 Venus 设备作为整个系统的主时钟源;
- 将其 10 MHz 时钟和同步信号（设备上电或复位后会自动产生）输出给第二个设备，以此类推，形成串接链;
- 后续设备的 TDC 时钟源设置为 “外部” 模式;
- 同步触发： 用户在软件中对主设备（第一个设备） 执行 TDC 复位操作，该复位命令将通过 sync_out 传递，致使链上所有设备的 TDC 同步复位。

二、安装与校准注意事项

- 线缆要求： 连接各设备的时钟和同步信号的同轴电缆应尽量等长，以最小化传输延迟差异;
- 系统标定： 尽管已使用等长电缆，但仍需对线缆延迟带来的固定时间差进行系统标定。标定得到的延迟值可输入至各设备的 INTER_DEALY 查找表中，软件将在数据处理时自动进行补偿;
- 设备标识： 用户可为每个 Venus 设备分配唯一的板号。该板号将作为标识信息被记录在 Raw data 中，便于后续的数据区分与融合处理。

根据第 10.12 节的测试结果，使用外部时钟源不会影响设备时间测量性能。

11.2 通过 API 接口进行多设备同步采集

11.2.1 通过以太网和交换机实现组网测试

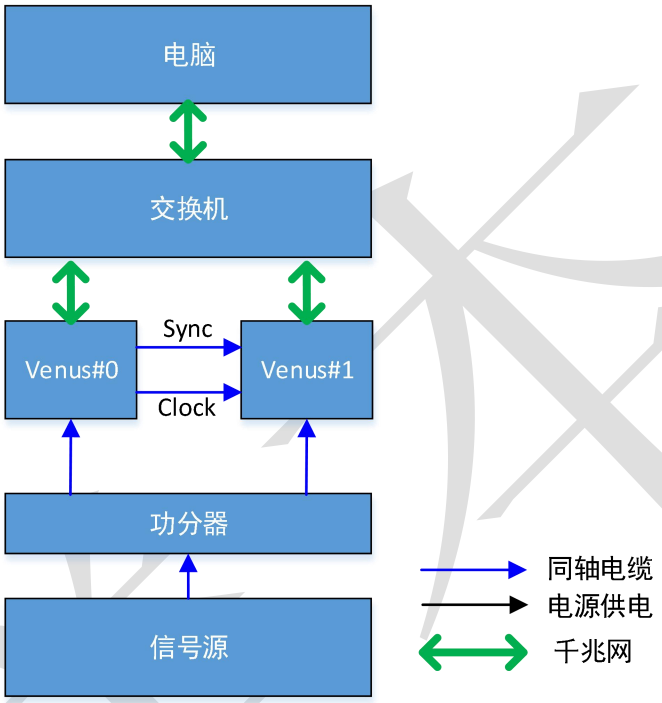


图 105 Venus 多设备组网测试示意图

Venus 支持多设备组网。根据 11.1 节介绍，Venus 组网有两种方式，本次测试以 Venus0 设备本地时钟作为全局时钟，Venus1 设备使用来自 Venus0 输出的时钟。信号源输出脉冲信号（上升沿/下降沿 3ns，幅度 4 V，重复频率 1 MHz），通过功分器将信号分为两路，分别接入 Venus0 设备的通道 0 和 Venus1 设备的通道 0,。设备输出的测量数据通过交换机（1G 或者 10G 交换机，本次测试仅介绍 1G 交换机为例，10G 组网类似操作）将数据传给上位机电脑。Venus 设备的配置参数见下表所示。注意，两台设备的 IP 地址和 MAC 地址需要区分。

表格 14 Venus 多设备组网测试设置参数

参数	Venus0	Venus1	本机电脑
IP 地址	10.0.0.10	10.0.0.20	10.0.0.5
端口地址	0x1234	0x1234	0x1234
MAC 地址	0x0123456789AB	0x0123456789CD	/
板号	0x0	0x1	
TDC 时钟来源	本地时钟	外部时钟	

官方提供基于 MATLAB 和 Python 的参考脚本，名称为 tdc_networking_example.m 和 tdc_networking_example.py。用户可以在此基础上进行二次开发和测试。该脚本基本的工作过程如下所示。

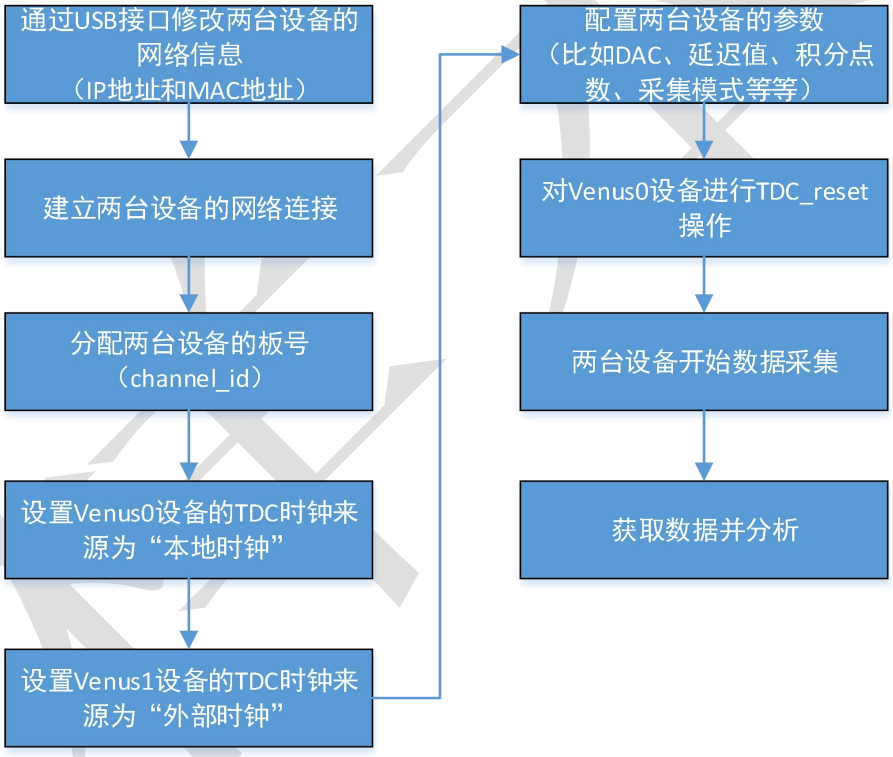


图 106 tdc_networking_example 脚本运行过程说明

具体说明如下。

（1） 首先，通过 USB 接口修改两台设备的网络信息。因为修改网络信息后可能引发设备网络暂时中断，通过 USB 接口设置比较稳妥。修改完网络信息后，可能需要重启交换机操作；

（2） 建立两台设备的网络连接；

（3） 分配两台设备的板号。两台设备的板号设置不同，以区分两台设备的 Raw

data；

（4） 设置 Venus0 设备的 TDC 时钟来源为“本地时钟”，设置为 Venus1 设备的 TDC 时钟来源为“外部时钟”。在本次测试中，系统 TDC 工作时钟来源于 Venus0 设备，因此需要设置 Venus0 设备的 TDC 时钟来源为“本地时钟”。如果用户通过其他时钟源来对 Venus0 提供 TDC 工作时钟的话，Venus0 需要配置为“外部时钟”；

（5） 配置 Venus0 和 Venus1 设备的采集参数，包括 DAC 阈值、时间延迟值、死时间值、积分点数、采集模式、采集时间等等信息；

（6） 对 Venus0 设备进行 TDC Reset，Venus0 设备会完成 TDC 计数值清零，并将同步清零信号传递给 Venus1 设备，Venus1 设备同步完成 TDC 计数值清零；

（7） 开启两台设备的数据采集，得到 Raw 数据文件。

得到两台设备的 Raw 数据，然后进行离线符合判选，得到 Venus0 的通道 0 与 Venus1 的通道 0 的符合时间分布，并分析其时间晃动值。

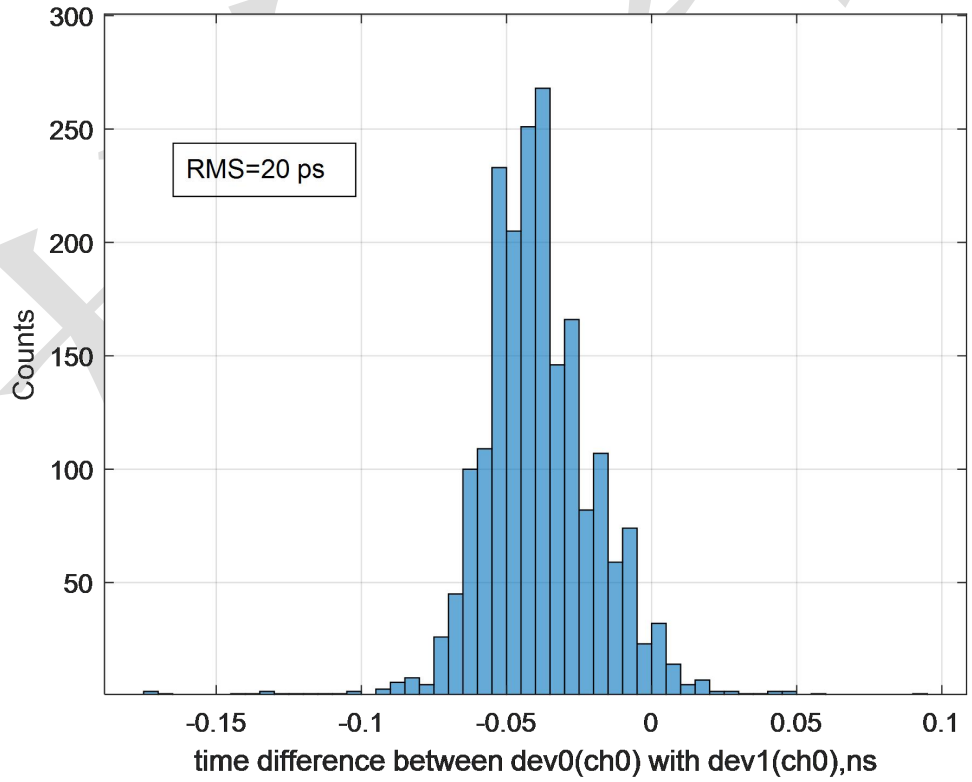


图 107 两个设备组网后各自通道间的时间晃动典型测试结果

12. 测量类型原始数据结构

12.1 数据包整体格式说明

Venus 分为 7 种测量数据，用户可以操作上位机软件中任意选择某一种测量类型。用户可以使用官方提供的 MATLAB 数据分析脚本来进行原始数据的转化，也可以自行对原始数据进行分析。

Venus 原始数据包由三部分组成，包头（64'hFFFFFFFFFFFFFFFF），地址字，32 组数据字和包尾（64'hFFFFFFFFFFFFFFFE）。一次测量数据由许多个这样的数据包组成。具体描述如下所示。

表格 15 测量裸数据包格式说明

包头（64'hFFFFFFFFFFFFFFFF）						
[63:60] 数据类型 0x0:Raw data	[59:52] 板号	[51:44] 默认 0x0:Raw data	[35:32] Reserve 0x0	[31:18] Product serial ID	[17:8] Firmware version	[7:0] FPGA core temperature, 0.5°C/lsb
Raw data 0						
Raw data 1						
……						
Raw data 31						
包尾（64'hFFFFFFFFFFFFFFFE）						

不同的裸数据类型，Raw data 格式不同。下面分别描述一下各种 Raw data 的数据结构和解析说明。

12.2 时间参考/全局符合数据

表格 16 时间参考/全局符合数据格式说明

RT [63:60]：0x4/5	保留 [59:40]	CH_ID-A [39:32]:通道号	CH_ID-B [31:24]:通道号	Tdiff(TB-TA) [23:0]:时间差 LSB = 0.975 ps

根据 RT 判断是合理的符合数据包，根据 CH_ID 筛选用户所选的通道号，提取 Tdiff (TB-TA)值。时间差根据以下公式计算。

$$t = Tdiff \times 0.975\ ps$$

注意时间差的正负，举例说明：

（1） 第一帧数据为 0x5_00000_00_01_000005;

表示 CH1 比 CH0 晚（大）5 LSB，即 delt_T(1,0) = 5 LSB；

（2） 第二帧数据为 0x5_00000_01_00_000005;

表示 CH1 比 CH0 早（小）5 LSB，即 delt_T(1,0)= - 5 LSB；

12.3 ADC 测量数据

表格 17 ADC 测量数据格式说明

RT [63:60]：0x2	Reserve [59:40]	CH_ID-A [39:32]:通道号	AD-A 值 [31:20]	CH_ID-B [19:12]:通道号	AD-B 值 [11:0]

首先根据 RT 判断是合理的能谱数据包，根据 CH_ID 筛选用户所选的通道号，然后提取 ADC 值。

12.4 能谱测量数据

表格 18 能谱测量数据格式说明

RT [63:60]=0x3	CH_ID [59:52]	能量值 [11:0]

首先根据 RT 判断是合理的能谱数据包，根据 CH_ID 筛选用户所选的通道号，然后提取能量值。

12.5 脉宽测量数据

表格 19 脉宽测量数据格式说明

RT [63:60]=0x6	Reserve [59:35]	CH_ID [34:28]:通道号	TOT 值 [27:0]：脉宽，LSB=0.975ps

根据 RT 判断是合理的脉宽测量数据包，根据 CH_ID 筛选用户所选的通道号，提取 TOT 值。脉宽值根据以下公式计算

$$t = TOT \times 0.975\ ps$$

12.6 时间-能谱全局符合数据

表格 20 时间-能谱全局符合测量数据格式说明

RT [63:60]=0x7	CH_ID-A [59:52]: 通道号	CH_ID-B [51:44]: 通道号	Area-A [43:30]: 积分面积	Area-B [29:16]： 积分面积	Tdiff(TA-TB) [15:0]:时间差 LSB = 0.975 ps

根据 RT 判断是合理的符合数据包，根据 CH_ID 筛选用户所选的通道号，提取 Tdiff (TA-TB)值。时间差根据以下公式计算。

$$t = Tdiff \times 0.975\ ps$$

12.7 双边沿时间戳数据

表格 21 双边沿时间戳测量数据格式说明

RT [63:60]=0x8	CH_ID [59:52]: 通道号	[51]: 边沿类型 0:上升沿 1:下降沿	时间戳： [50:0] LSB = 0.975 ps

根据 RT 判断是合理的符合数据包，根据 CH_ID 筛选用户所选的通道号，提取上升沿（[51]:0）和下降沿([51]:1)的时间戳信息。

$$t = T \times 0.975\ ps$$

12.8 周期测量数据

RT [63:60]=0x9	CH_ID [59:52]: 通道号	[51]: 边沿类型 0:下降沿-上升沿时间 1:周期时间	时间戳： [50:0] LSB = 0.975 ps

根据 RT 判断是合理的符合数据包，根据 CH_ID 筛选用户所选的通道号，提取下降沿-上升沿时间（[51]=0x0 时的[50:0]值，t_1）和周期时间（[51]=0x1 时的[50:0]值,t_2）。

$$t = T \times 0.975\ ps$$

因此，占空比为 t_1/t_2 x 100%。

13. 支持定制化和二次开发

Venus 支持用户对硬件、下位机固件和上位机软件进行定制或者二次开发。具体如下所示。

表格 22 官方支持的定制化和二次开发服务

部分	支持定制	支持二次开发	说明
硬件	模拟前端		具体联系厂商
	存储器		
	对外接口		
下位机固件	TDC 分辨率	AUX 对外接口可以开放 JTAG 信号，开放 FPGA 工程，用户可以直接进行二次开发。	具体联系厂商
	TDC 死时间		
	数据格式		
	对外接口协议		
	其他功能		
上位机软件	GUI 分析功能		具体联系厂商
	基于 API 接口的应用开发		
	MATLAB/Python 裸数据分析脚本		

附录

A. 官方提供设备和附件列表

表格 23 Venus 官方提供设备和附件列表

序号	设备/资料名称	数目	基础/选配	说明
1.	Venus 设备	1	基础	
2.	数据手册	1	基础	
3.	电源适配器/线	1	基础	
4.	USB3.0 线缆	1	基础	
5.	千兆网线	1	基础	
6.	AUX 对插接口	1	基础	
7.	SMA 公母转接头	2	基础	
8.	上位机软件安装包	1	基础	
9.	API 接口函数包	1	基础	
10.	MATLAB/Python 数据分析脚本	1	基础	
11.	万兆网卡+光纤转接器	1	选配	需额外付费
12.	定制化需求	1	选配	需额外付费

B. 固件版本说明

Venus lite 中的 TDC 采取了多链 FPGA-TDC 技术，因此可以实现无需硬件改动的升级任务。目前，有如下固件版本可以选择，具体请联系厂商咨询。随着产品继续迭代，我们可能会随时进行更新。

表格 24 Venus lite 下位机固件可选说明

下位机固件版本	支持设备	主要性能	说明
V1	Venus 系列	CH0-CH31:普通通道 （8 ps RMS, 4 ns 死时间）	默认版本
V3		CH0-CH3: 超高精度通道 （4 ps RMS, 4 ns 死时间） CH4-CH16:普通通道 （8 ps RMS, 4 ns 死时间）	默认版本
V4		CH0-CH3: 超高精度通道	默认版本

	Venus lite 系列	（3 ps RMS, 2 ns 死时间） CH4-CH16:普通通道 （6 ps RMS, 2 ns 死时间）	

C. 硬件和软件版本说明

下表描述了 Venus 硬件和软件版本迭代/进化说明。

表格 25 硬件版本进化说明

硬件版本	发布时间	主要改动	说明
V1.1	2025.10	原型样机	
V1.2	2026.2	量产机版本	1. 增加了迟滞电压调节； 2. 优化了电源和时钟电路；

表格 26 软件版本进化说明

软件版本	发布时间	主要改动	说明
V1.0.0	2026.2	初版	

D. 对外数据接口列表

表格 27 对外数据接口列表

序号	接口名称	管脚解释		说明
1.	AUX	1	机壳地	
		5,6,7,13,14,15	数字地	
		2,3,4	12V 供电输入	
		8	3.3V	
2.	千兆以太网			RJ-45
3.	USB3.0			USB-A

4.	QSFP 接口	4 路光口，默认第 1 路为万兆网口。	用户需要 QSFP 转 SFP 光纤收发器转接
5.	电源入口	+12 V	EJ508A

E. 时间晃动分析模型

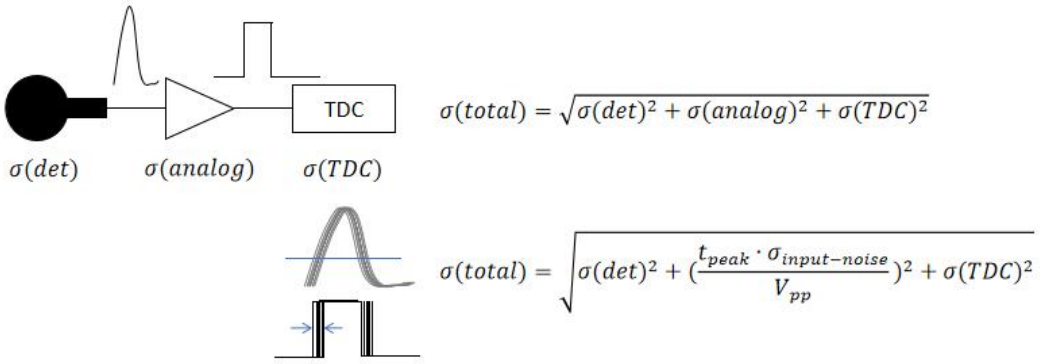


图 108 一般测量系统时间晃动分析模型

探测器输出的脉冲信号经过 Venus 系统模拟前端电路后进入 TDC 测量系统。假设探测器输出信号本身的时间晃动（即待测量值，RMS）为σ(det)，模拟前端电路因为电压噪声引入的时间晃动为σ(analog)，TDC 分辨率造成的时间晃动为σ(TDC)，那么总的时间晃动如下所示：

$$\sigma(total) = \sqrt{\sigma(det)^2 + \sigma(analog)^2 + \sigma(TDC)^2}$$

其中，电路电压噪声引入的时间晃动σ(analog)与输入信号的斜率 k 和电路输入噪声σ(input-noise)相关。而信号斜率 k 与探测器响应时间、电路带宽、摆率和寄生参数相关。我们假设输入信号达峰时间为 t_{peak}，脉冲电压峰值为 V_{pp}，那么总到时间晃动可以近似为：

$$\sigma(total) = \sqrt{\sigma(det)^2 + (\frac{t_{peak} \cdot \sigma_{input-noise}}{V_{pp}})^2 + \sigma(TDC)^2}$$

我们可以根据本公式来近似推算总到时间晃动值，以及选择的测量系统是否可以满足我们到测量需求。

比如，我们让 Venus 进入内部测试模式，可以估算σ(TDC)，比如根据 10.2 节，对于 **Venus Lite** 系列的高分辨率通道，TDC 本身的时钟晃动约为 3 ps RMS；对于 **Venus Lite** 系列

的普通测量通道，TDC 本身的时钟晃动约为 6 ps RMS；

然后，我们根据 Venus 到系统标定模式，根据 10.1 节，可以得到每个通道到输入噪声电压约为 2 mV RMS；

假设，我们输入了 600 mV 幅度信号，达峰时间（注意不是上升时间）约为 2 ns，那么根据公式可以得到 **Venus Lite** 系统到高分辨率通道引入的时间晃动约为$\sqrt{9+49}=7.6\ ps\ RMS$，普通通道引入到时间晃动约为$\sqrt{36+49}=9.2\ ps\ RMS$

F. 在线符合逻辑说明

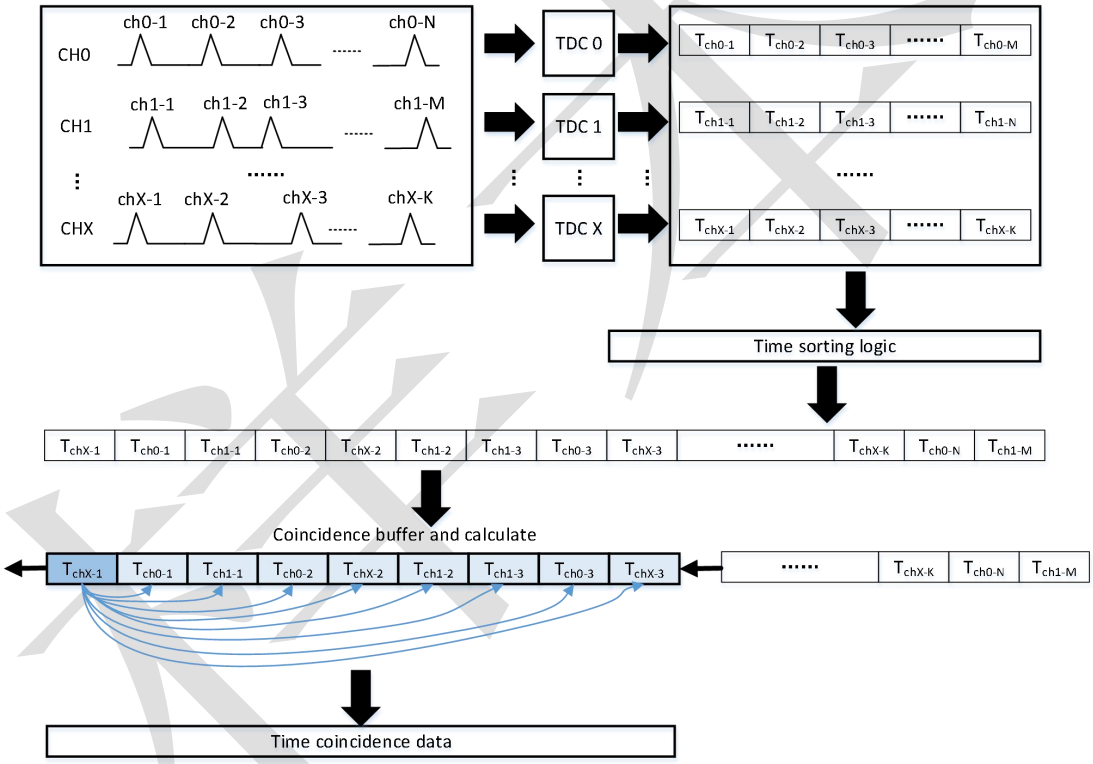


图 109 Venus 在线时间符合事件判断逻辑说明

Venus 设备采用基于 FPGA 的硬件在线符合事件筛选逻辑，以保证符合筛选的可靠性和效率。其基本过程如上图所示。首先，各个 TDC 通道计算各个通道的输入信号到达时间；然后，根据事件的时间信息按照先后顺序进行排序；然后，依次将排序后的时间数据输入到符合缓存区，并计算第一个事件的时间与缓存区内的其他事件的时间进行符合判选，在时间符合窗内的事件保留并计算时间差；然后下一个数据进入缓存区，重复以

上过程。因此，此符合判选过程会计算所有在符合时间窗内的事件，并输出给上位机软件。

G.归一化自相关系数和时钟偏差计算公式

Venus 设备中采用的修正艾伦偏差（MDEV）、哈达玛偏差（HDEV）、时间偏差（TDEV）和艾伦偏差（ADEV）的计算公式如下所示。其中，x 为周期测量结果，τ为分析时间，N 为统计点数。

$$ADEV = \sqrt{\frac{1}{2(N-2\tau)\tau^2}\sum_{i=1}^{N-2\tau}(x_{i+2\tau}-2x_{i+\tau}+x_i)^2}$$

$$MDEV = \sqrt{\frac{1}{2\tau^4(N-3\tau+1)}\sum_{j=1}^{N-3\tau+1}\left[\sum_{i=j}^{j+\tau-1}(x_{i+2\tau}-2x_{i+\tau}+x_i)\right]^2}$$

$$TDEV = \frac{\tau}{\sqrt{3}}\sqrt{\frac{1}{2(N-2\tau)\tau^2}\sum_{i=1}^{N-2\tau}(x_{i+2\tau}-2x_{i+\tau}+x_i)^2}$$

$$HDEV = \sqrt{\frac{1}{6\tau^2(N-3\tau)}\sum_{i=1}^{N-3\tau}(x_{i+3\tau}-3x_{i+2\tau}+3x_{i+\tau}-x_i)^2}$$

Venus 中归一化自相关系数计算公式如下所示。其中，x 为双通道时间差测量结果，k 为分析阶数，N 为统计点数。

$$AC(k) = \frac{\sum_{t=1}^{N-k}[x_t - mean(x)][x_{t+k} - mean(x)]}{\sum_{t=1}^{N}[x_t - mean(x)]^2}$$

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