Chronos

Venus lite series.
16/8 Channel, 0.976 ps
Streaming Time-Digital Converter

Venus Lite User Guide

UG-01-3-054-1, Nov. 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

Copyright Notice

Copyright © 2026 by Chronos Technology Inc. All rights reserved.
This document contains information that is proprietary and confidential to Chronos Technology Inc. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher.

目录

1. 产品特征 14
2. 技术规格 15
3. 产品订购选型 20
4. 产品功能结构和应用场景 20
5. 产品外观尺寸 22
6. 绝对最大额定值 23
7. 设备接地和静电 24
    7.1 地环噪声的影响 24
8. 上位机软件使用说明 25
    8.1 软件安装说明 25
        8.1.1 Windows 系统 25
        8.1.2 Linux 系统 32
    8.2 上位机电脑硬件要求 34
    8.3 接口配置 35
        8.3.1 千兆网口 35
        8.3.2 万兆网口 37
        8.3.3 USB 3.0 39
    8.4 界面介绍 40
    8.5 登录页面 41
    8.6 用户页面 41
        8.6.1 通道使能 42
        8.6.2 比较阈值配置 43
        8.6.3 各通道迟滞电压配置 44
        8.6.4 各通道延迟配置 45
        8.6.5 各通道死时间配置 46
        8.6.6 ADC 积分点数 48
        8.6.7 通道参数快速设置（One-Key） 49
        8.6.7 信号边沿选择 50
        8.6.8 采集模式选择 52
        8.6.9 测量数据保存 61
    8.7 分析页面 63
        8.7.1 强度分析 64
        8.7.2 双通道时间差分布直方图 65
        8.7.3 Start-Stop 分组测试 69

8.7.4 ADC 波形 73
8.7.5 能谱分布 74
8.7.6 TOT 脉宽分布 75
8.7.7 符合能谱-时间谱分析 77
8.7.8 频率分析 79
8.7.9 二阶自关联函数（ $g^{(2)}$ ）分析 81
8.7.10 TDC 非线性测试 83
8.7.11 多重符合统计 84
8.7.12 分组二阶互关联系数（g_{AB}^{2}）..... 88
8.7.13 FLIM 扫描测试 89
8.9 管理页面 93
8.9.1 基本介绍 93
8.9.2 辅助 Gate 输出 95
8.10 日志区 97
9. 开发者 API 接口 97
9.1 C API 97
9.1.1 概述 97
9.1.2 初始化和销毁 98
9.1.3 设备连接 98
9.1.4 错误处理 100
9.1.5 设备自检 101
9.1.6 通道配置 102
9.1.7 全局配置 105
9.1.8 状态查询 111
9.1.9 网络配置 113
9.1.10 数据采集 115
9.1.11 数据分析 117
9.1.12 其他辅助接口 132
9.1.13 通用注意事项 133
9.2 C++ API 133
9.2.1 概述 133
9.2.2 初始化与设备连接 134
9.2.3 设备自检 136
9.2.4 通道配置 136
9.2.5 全局配置 140
9.2.6 网络配置 149

9.2.7 数据采集 152
9.2.8 数据分析 155
9.2.9 注意事项 174
9.2.10 示例代码 174
9.3 PYTHON API 176
9.3.1 概述 176
9.3.2 初始化 176
9.3.3 设备连接 176
9.3.4 错误处理 177
9.3.5 设备自检 178
9.3.6 通道配置 179
9.3.7 全局配置 181
9.3.8 状态查询 186
9.3.9 网络配置 187
9.3.10 数据采集 188
9.3.11 数据分析 190
9.3.12 注意事项 203
9.3.13 示例代码 203
9.4 MATLAB API 206
9.4.1 概述 206
9.4.2 初始化 206
9.4.3 设备连接 206
9.4.4 错误处理 207
9.4.5 设备自检 208
9.4.6 通道配置 208
9.4.7 全局配置 210
9.4.8 状态查询 215
9.4.9 网络配置 217
9.4.10 数据采集 218
9.4.11 数据分析 219
9.4.12 注意事项 230
9.4.13 示例代码 230
9.5 LabVIEW API 235
9.5.1 概述 235
9.5.2 环境准备 235
9.5.3 初始化和销毁 236

9.5.4 设备连接 237
9.5.5 设备自检 238
9.5.6 通道配置 238
9.5.7 全局配置 243
9.5.8 状态查询 250
9.5.9 网络配置 252
9.5.10 数据采集 255
9.5.11 数据分析 259
9.5.12 注意事项 275
9.5.13 参考示例 276
9.6 原始数据离线分析参考（Matlab） 277
9.6.1 MATLAB 原始数据分析参考脚本 277
9.6.2 MATLAB 分析脚本组成 278
9.6.3 分析程序使用方法 278
9.6.4 运行举例 279
9.7 原始数据离线分析参考（Python） 286
10. 典型测试电路和测试结果 288
10.1 测试简介 288
10.2 双通道时间差测量（内部触发） 288
10.2.1 标准配置 288
10.2.2 2ps 订制版本测试 290
10.3 双通道时间差测量（外部触发） 292
10.3.1 标准配置 292
10.3.2 2ps 订制版本测试 295
10.4 单通道重复频率测量 296
10.5 单通道脉宽测量 298
10.6 模拟-数字转换实时波形测量 300
10.7 能谱测量 304
10.8 全局时间-能谱符合测量 306
10.9 死时间配置测试 307
10.10 环境温度与 FPGA 核心温度关系 308
10.11 时间晃动的温漂（内部触发） 310
10.12 各通道与参考通道延迟温漂 311
10.13 内外部时钟与时间晃动 312
10.14 时间晃动稳定性 313
10.15 能量测量线性度和精度 313

10.16 硬件在线 N 重符合测试结果 315
11. 多设备时钟同步 319
11.1 多设备时钟同步 319
11.2 通过 API 接口进行多设备同步采集 321
11.2.1 通过以太网和交换机实现组网测试 321
11.2.2 通过 USB 实现组网测试 324
11.3 并行数据获取方案 326
12. 测量类型原始数据结构 327
12.1 时间戳数据 328
12.2 时间参考/全局符合数据 329
12.3 ADC 测量数据 330
12.4 能谱测量数据 330
12.5 脉宽测量数据 330
12.6 时间-能谱全局符合数据 331
12.7 双边沿时间戳数据 331
12.8 周期测量数据 331
12.9 带时间戳的全局符合数据 332
12.10 单边沿时间戳数据 332
12.11 时间全局符合数据 2 333
13. 支持定制化和二次开发 334
14. 产品校准与维护 335
15. 常见问题答疑 336
附录 346
A. 官方提供设备和附件列表 346
B. 固件版本说明 346
C. 硬件和软件版本说明 347
D. 对外数据接口列表 348
E. 时间晃动分析模型 348
F. 在线 2 重符合逻辑说明 350
G. 归一化滞后自相关系数和时钟偏差计算公式 351
H. 二阶自关联函数（g^2）算法 352
I. 二阶互关联函数（g^2_{AB}）算法 352
J. 通道间延迟值对齐方法 353
本文档历史记录 355

图目录

图 1 Venus 系列产品功能结构框图 20
图 2 几种典型的应用实验场景 21
图 3 Venus Lite-T 系列产品外观和结构尺寸图 22
图 4 Venus Lite 系列产品外观和结构尺寸图 23
图 5 基于 USB3.0 接口的测试系统地环路噪声问题示意图 24
图 6 上位机安装步骤 2 26
图 7 上位机安装步骤 3 26
图 8 上位机安装步骤 4 27
图 9 上位机安装步骤 4 安装过程 27
图 10 上位机安装完成 28
图 11 USB3.0 驱动设备管理器 29
图 12 手动安装 USB3.0 驱动 31
图 13 手动安装 VC++运行库 32
图 14 软件安装 Linux 命令 33
图 15 Linux 系统下的软件运行 34
图 16 上位机电脑千兆网网络设置 36
图 17 网口 IP 和端口地址修改页面 37
图 18 万兆网物理连接示意图 38
图 19 上位机电脑万兆网网络设置 38
图 20 网口 IP 和端口地址修改页面 39
图 21 USB3.0 驱动安装信息 40
图 22 登录页面 41
图 23 用户页面 42
图 24 通道使能勾选 43
图 25 模拟信号比较功能示意图 43
图 26 触发阈值配置 GUI 示意图 44
图 27 迟滞电压原理示意图 44
图 28 比较器迟滞电压配置 GUI 示意图 45
图 29 通道间延迟配置 GUI 示意图 46
图 30 设置死时间以消除“振铃”带来的无用数据读出 47
图 31 通道时间测量死时间配置 GUI 示意图 47
图 32 能谱测量数据模式逻辑框图 48
图 33 A/D 通道采样点个数配置 GUI 示意图 48
图 34 One-key 配置方式 49
图 35 CHANNELS_CONFIG.txt 文件格式要求 50
图 36 CHANNELS_CONFIG.txt 文件保存当前配置 50
图 37 边沿采集模式配置 GUI 示意图 51

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。）..... 51
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。）.....52
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。）.....52
图 41 两种采集启动方式 53
图 42 使用外部硬件采集脉冲实现多设备高精度同时采集 54
图 43 数据采集模式选择 55
图 44 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。.....56
图 45 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。.....57
图 46 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）... 58
图 47 ADC 测量数据模式逻辑框图 58
图 48 能谱测量数据模式逻辑框图 59
图 49 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。 59
图 50 测量数据保存配置 62
图 51 数据保存格式 62
图 52 各通道强度（计数率）分析（实时计数率和符合计数率）.....64
图 53 双通道时间差分布直方图 65
图 54 自定义拟合函数键入窗口 67
图 55 双通道互关联系数实时计算 67
图 56 双通道时间差分析过程中 T3 数据的保存 68
图 57 Start-stop 分组测试界面 69
图 58 Start-stop 统计示意图（以上升沿触发为例，以 CH0 作为 start 通道，CH1/2/3 为 Stop 通道为例。实际用户可以按照实验要求自行分组）.....70

图 59 Start-stop 输出 T3 数据的格式示意图（包含 Start 触发序列、start 通道号、stop 通道号、时间差信息）.....71
图 60 双通道 Stop-Start 时间差热点图.....72
图 61 一种常见的 Start-stop 功能应用测试场景.....72
图 62 ADC 瞬态波形分析.....73
图 63 能谱分布分析.....74
图 64 TOT 脉宽分布分析.....75
图 65 TOT 分析过程数据保存.....76
图 66 符合能谱-时间谱分析.....77
图 67 一种典型的能谱-时间符合测试平台.....78
图 68 单通道频率分析.....79
图 69 二阶自关联函数分析.....81
图 70 保存 T2 时间戳数据.....82
图 71 TDC 非线性测试.....83
图 72 两种 N 重符合机制.....84
图 73 N 重符合上位机软件示意图.....85
图 74 策略配置文件.....85
图 75 统计结果保存格式.....86
图 76 多重符合示意图（以 2 个策略组，3/4 重符合为例）.....86
图 77 符合分析结果是否保存以及保存文件格式.....87
图 78 分组二阶互关联系数计算.....88
图 79 二阶互关联分析中过程 T3 数据保存.....88
图 80 FLIM 扫描软件功能.....89
图 81 同步扫描位置信号（Frame/Line/Pixel 示意图，以 2 帧，8x8 图像为例）.....90
图 82 FLIM 定位扫描模式下的数据输出格式.....91
图 83 FLIM 非定位模式.....91
图 84 FLIM 非定位扫描模式下的数据输出格式.....92
图 85 FLIM 中 T2 模式.....92
图 86 FLIM 中 T2 模式下的数据输出格式.....93
图 87 管理页面.....93
图 88 Gate 输出功能选择.....95
图 89 系统日志.....97
图 90 数据分析 MATLAB 脚本运行.....279
图 91 选择要分析的 adc Raw 数据文件.....280
图 92 数据分析 MATLAB 脚本运行.....280
图 93 输出“adc.txt”数据格式.....281
图 94 输出“tot.txt”数据格式.....281
图 95 输出“area.txt”数据格式.....282

图 96 输出 “global_coins.txt” 数据格式 282
图 97 输出 “area_coins.txt” 数据格式 283
图 98 输出 “dual_singles.txt” 数据格式 283
图 99 输出 “period.txt” 数据格式 284
图 100 输出” glbt_coins.txt” 数据格式 284
图 101 输出” sig_singles.txt” 数据格式 285
图 102 Channel_delay_lut 输出 285
图 103 Raw 数据分析 python 脚本分析过程 287
图 104 双通道时间差测量示意图（内部触发） 288
图 105 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, CH0/2) 289
图 106 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，CH4/5） 289
图 107 Venus 设备质量检验报告-参考样本中的内部触发测试结果(没有除以√2) 290
图 108 Venus Lite4-2ps 特殊版本双通道时间晃动测试（内部触发，上图：通道 0 和通道 1，下图：通道 2 和通道 3） 291
图 109 双通道时间差测量示意图 292
图 110 双通道时间差外部输入信号测量结果（高分辨率通道，通道 0 和 1，上升沿，典型结果） 293
图 111 双通道时间差外部输入信号测量结果（普通通道，通道 8 和 9，上升沿，典型结果） 293
图 112 Venus 设备质量检验报告-参考样本中的外部触发测试结果(没有除以√2) 295
图 113 Venus Lite4-2ps 特殊版本双通道时间晃动测试（外部触发，上图：通道 0 和通道 1，下图：通道 2 和通道 3） 296
图 114 单通道周期信号重复频率测量示意图 296
图 115 单通道重复频率测量 297
图 116 时钟方差和相位误差 297
图 117 单通道信号脉宽测量示意图 298
图 118 TOT 外部输入信号测量结果（高分辨率通道，通道 0，典型结果） 299
图 119 TOT 外部输入信号测量结果（普通通道，通道 4，典型结果） 299
图 120 Venus 设备质量检验报告-参考样本中的外部触发 TOT 测试结果 300
图 121 模拟-数字转换实时波形测量 300
图 122 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） 301
图 123 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） 302
图 124 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） 303

图 125 能谱测量示意图 304
图 126 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） 305
图 127 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） 306
图 128 全局时间-能谱符合测量示意图 306
图 129 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） 307
图 130 温度测试平台示意图 308
图 131 温度循环实验温度曲线设置 309
图 132 环境温度与 FPGA 内核稳定温度典型测试结果（Venus lite-T） 310
图 133 双通道时间差晃动与环境温度关系测试典型结果 311
图 134 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） 312
图 135 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） 313
图 136 能谱测量示意图 313
图 137 能量测量线性和分辨率测试典型结果（通道 1） 314
图 138 N 重符合测试平台示意图 315
图 139 N 重符合计数率统计损失典型测试结果（硬件符合模式，40 组符合策略同时运行。上图：输入计数率与单计数率和某策略组符合计数率关系；下图：输入计数率与某策略组符合计数率损失关系） 316
图 140 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） 319
图 141 Venus 多设备组网测试示意图（以太网） 321
图 142 tdc_networking_example 脚本运行过程说明 322
图 143 双板通道间的时间差分布直方图典型测试结果 324
图 144 tdc_networking_example 脚本运行过程说明 325
图 145 Venus 多设备组网测试示意图（USB） 326
图 146 一种可能的并行数据获取系统架构 326
图 147 双通道时间差与 Start-stop 测试区别示意图（上升沿触发举例。 344
图 148 一般测量系统时间晃动分析模型 348
图 149 Venus 在线时间符合事件判断逻辑说明 350

表目录

表格 1 Venus 系列系统技术规格参数 15
表格 2 Venus 系列产品订购选型 20
表格 3 Venus 设备工作额定值 23
表格 4 上位机电脑配置需求 34
表格 5 裸数据采集模式说明 60
表格 6 上位机软件用户界面各个输入参数说明 62
表格 7 设备序列号解释 94
表格 8 Gate 输出功能描述 95
表格 9 MATLAB 原始数据分析脚本输出结果说明 277
表格 10 Raw 数据分析参考脚本说明 286
表格 11 某机器典型时间分辨率测试结果（内部触发，典型结果） 290
表格 12 某机器典型时间分辨率测试结果（外部触发） 294
表格 13 TOT 外部输入信号典型测量结果 300
表格 14 不同死时间与 OCR 关系测试结果（ICR 为 20 Mcps，通道 0） 308
表格 15 硬件 N 重符合性能参数 316
表格 16 Venus 多设备组网测试设置参数 322
表格 17 测量裸数据包格式说明 327
表格 18 Time zone 数据格式说明 328
表格 19 Event 数据戳格式说明 328
表格 20 一个时区内，只有一个时间戳发生时在另一个时间戳内全部填充 1 329
表格 21 时间参考/全局符合数据格式说明 329
表格 22 ADC 测量数据格式说明 330
表格 23 能谱测量数据格式说明 330
表格 24 脉宽测量数据格式说明 330
表格 25 时间-能谱全局符合测量数据格式说明 331
表格 26 双边沿时间戳测量数据格式说明 331
表格 27 周期测量数据格式说明 331
表格 28 带时间戳的全局符合数据格式说明 332
表格 29 单边沿时间戳测量数据格式说明 332
表格 30 时间参考/全局符合数据格式说明 333
表格 31 官方支持的定制化和二次开发服务 334
表格 32 Venus 官方提供设备和附件列表 346
表格 33 硬件版本进化说明 347
表格 34 软件版本进化说明 347
表格 35 对外数据接口列表 348

1. 产品特征

功能丰富



- 16/8 通道时间-数字转换器（TDC）
- 8/4 通道模拟-数字转换器（ADC）
- 边沿定时电路，高精度阈值电压设置
- 多种裸数据存储/测量/在线分析模式：计数率/流式时间戳/在线全局符合/脉宽/时钟稳定性分析/能谱/波形测量等
- 支持单光子计数统计模式（TCSPC），Multi-start/stop 分组测试
- 支持硬件/软件在线分组多重符合统计
- 支持二阶（自/互）关联函数实时计算
- 支持 IEEE 1139 时钟稳定性分析
- 支持 FLIM 成像数据扫描
- 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）
- 外部输入时钟频率 10/25 MHz 可选
- 支持软件采集和硬件触发采集模式
- USB3.0(70 Mtags/s)/1G 以太网(25 Mtags/s)/10 G 以太网(180 Mtags/s)数据接口、提供 40 Gbps QSFP(720 Mtags/s)等定制接口协议
- 配套数据采集上位机软件（Windows/Linux），配套数据处理软件包
- 免费开放 C/C++/MATLAB/Python/LabVIEW API 接口和应用示例
- +12 V 直流输入供电

高性能



- 0.976 ps 时间测量数字分辨率
- 单通道时间测量精度~ 3 ps RMS
- 通道时间延迟可配置，0 ~ 1 ms (LSB: 0.976 ps)
- 单通道时间测量死时间 2 ~ 100000 ns，可配置
- 硬件在线 N 重符合策略数 40，16 通道任意部署
- 频率分析范围 8 kHz - 500 MHz
- 附加 50 Msps, 12 bit ADC 采样
- 模拟前端 50 Ohms 输入阻抗
- 可设置甄别阈值范围-1.5 V - +1.5 V，14 bit
- 可设置甄别迟滞电压 1/30/40/70 mV

其他

- 多链 FPGA-TDC 技术

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 5 V，上升沿 3 ns，重复频率 1 MHz，经过功分器输入到 Venus 设备 16 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，则单通道时间晃动为σ/√2.

表格 1 Venus 系列系统技术规格参数

参数	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	被动散热	风冷		
风扇个数	/	2		-T 版本：
风扇转速	/	4300	RPM	风扇自动调速
气流	/	9.7 x 2	CFM	转速实时监控
噪声	/	22 x 2	dB	温度实时监控
测量通道数				
时间测量通道数	16 或 8	16 或 8		
AD 测量通道数	8 或 4	8 或 4	个	参考产品列表 [A]
TOT 测量通道数	16 或 8	16 或 8		
参考测量通道				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
通道数	1	1	个	
输入阻抗	50±1%	50±1%	Ω	
输入信号高电平	2.4/3.3	2.4/3.3	Vmin/max	
输入信号低电平	0/0.8	0/0.8	Vmin/max	
外部时钟输入				
标准输入频率	10/25	10/25	MHz	对于 V1.3 版本硬件，仅支持 25 MHz 时钟输入； 对于 V1.4+版本硬件，支持 10 MHz/25 MHz 时钟输入
输入信号高电平	2.4/3.3	2.4/3.3	Vmin/max	
输入信号低电平	0/0.8	0/0.8	Vmin/max	
外部同步输入				
最小高电平宽度	40	40	ns	

输入信号高电平	2.4/3.3	2.4/3.3	Vmin/max	高电平有效 [B]
输入信号低电平	0/0.8	0/0.8	Vmin/max	
系统输出时钟				
同步输出通道数	1	1	个	
输出时钟频率	25	25	MHz	
输入信号高电平	2.4/3.3	2.4/3.3	Vmin/max	
输入信号低电平	0/0.8	0/0.8	Vmin/max	
Gate 输入/输出				
输入信号高电平	2.4/3.3	2.4/3.3	Vmin/max	
输入信号低电平	0/0.8	0/0.8	Vmin/max	
脉冲延迟调整范围	0/2.5	0/2.5	ns	
脉冲延迟调整分辨	78	78	ps	
模拟前端 [C]				
增益	+1	+1	V/V	
输入阻抗	50±1%	50±1%	Ω	
输入电容	5	5	pF	
输入信号范围				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
正向脉冲	0/+2	0/+2	Vmin/max	
负向脉冲	-2/0	-2/0	Vmin/max	
可设阈值				软件限制阈值设置范围。如需要开放设置范围，请联系时溯科技官方。
范围	-1.5/+1.5	-1.5/+1.5	Vmin/max	
分辨率	0.6	0.6	mV	
TDC 参数				LSB is ~ 0.976 ps **注意**，LSB 准确的值应该是 0.9765625 (1000/1024) ps。本设备配套的上位机软件均采用精确值转换。如果用户需要精确离线分析数据，建议用户使用 0.9765625 ps。
测量最小刻度	0.976	0.976	ps	
在线非线性修正	是	是		
DNL(高分通道)	-1 ~ 10	-1 ~ 10	LSB	
DNL(普通通道)	-1 ~ 40	-1 ~ 40	LSB	
INL(高分通道)	<±20	<±20	LSB	
INL(普通通道)	<±40	<±40	LSB	

				为了简化文档编写，本文档近似为 0.976 ps。
ADC 参数				
采样率	50	50	Msps	
分辨率	12	12	Bits	
INL	<±1	<±1	LSB	Fin=10 MHz
DNL	<±0.65	<±0.65	LSB	
SNR	70	70	dB	
ENOB	11.3	11.3	bits	
SINAD	70	70	dB	
单通道时间测量 [D]				高分辨率通道指 Ch0-3 通道 内部触发测试
普通通道抖动	~6.0	~6.0	ps RMS, σ/√2	
高分辨率通道抖动	~3.0	~3.0	ps RMS, σ/√2	对于 4/5 通道 2ps 特殊定制产品，四/五个通道晃动可低至 2 ps RMS，参见 10.2.2 节。
单通道脉宽测量 [D]				高分辨率通道指 Ch0-3 内部触发测试
最小测量脉宽	0.5	0.5	ns	
最大测量脉宽	16	16	us	
时间测量死时间				
死时间可设范围	2/100000	2/100000	ns min/max	上位机软件可配置
死时间设置分辨	2	2	ns	
迟滞电压				上位机软件可配置
可设置档位	1/30/40/70	1/30/40/70	mV	
通道时间延迟				
通道延迟可设范围	0/1000000	0/1000000	ns	
通道延迟设置分辨	0.976	0.976	ps	
输出数据类型分类 [E]				
时间戳数据	支持	支持		用户可以采集/保存任一数据类型
双沿时间戳数据	支持	支持		

时间参考符合数据 时间全局符合数据 ADC 测量数据 面积测量数据 时间-能谱全局符合 过阈时间测量数据 周期测量数据 带时间戳符合数据	支持 支持 支持 支持 支持 支持 支持 支持	支持 支持 支持 支持 支持 支持 支持 支持		的裸数据，支持.bin/.hex 格式 用户可以使用裸数据自行进行分析以保证数据的高度可靠性
在线时间符合 符合时间半窗范围 符合时间半窗分辨	 0/16000 0.976	 0/16000 0.976	 ns min/max ps	全局时间符合窗。如果进行多重符合分析，可建立虚拟符合时间窗，参见 8.7.11 节。
在线能量积分 积分最大时间 积分时间分辨	 1.3 20	 1.3 20	 ms ns	
单通道统计计数率	<250	<250	Mtags/ch	
在线 N 重符合统计				软件 N 重符合性能依赖上位机电脑性能和数据流量
独立策略组数	40（硬件方式） 200（软件方式）	40（硬件方式） 200(软件方式)	组	
每组通道配置数	16	16	个	
每组虚拟时间窗	0/63	0/63	ns Min/max	
符合计数率统计	<100（硬件方式）	<100（硬件方式）	Mtags/s	
统计结果更新周期	1	1	s	
时钟稳定性分析 **输入时钟频率范围**	 0.008/500	 0.008/500	MHz min/max	
FLIM 扫描				
输入信号 **（定位模式）**	Frame/Line/Pixel 位置信号 Sync、Signal 信号	Frame/Line/Pixel 位置信号 Sync、Signal 信号		
输入信号 **（非定位模式）**	Sync、Signal 信号	Sync、Signal 信号		
扫描帧率	32 (CCP=1)	32(CCP=1)	FPS	USB3.0 接

				口,1024x1024 图像
在线事件缓存	250	250	Mtags	
对外数据接口带宽 [F]				1 tag 指 1 个时间戳或 ADC 波形数据事例 对于 V1.4+版本硬件，USB 支持 70 Mtags/s
USB 3.0	40/70	40/70	Mtags/s	
千兆以太网	25	25	Mtags/s	
万兆以太网	180	180	Mtags/s	
定制 4 路 QSFP	720	720	Mtags/s	
系统供电 [G]				
直流电压/电流	12/>3	12/>3	V/A	
直流供电纹波要求	<0.1	<0.1	Vpp	
系统功耗	<25	<25	W	
设备尺寸 [H]	190x170x42	190x190x73	WxLxH,mm^3	
外壳材料	铝 6061	铝 6061		
丝印油墨材料	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	通道时间晃动，RMS	AD 测量通道数	对外接口	供电方式
Venus[A]	Venus lite (-T) 16	16[B]	CH0-3: ~3 ps CH4-15: ~6 ps	8	USB3.0/千兆网/万兆网	12V 直流供电
	Venus lite (-T) 8	8[B]	CH0-3: ~3 ps CH4-7:~6 ps	4	USB3.0/千兆网/万兆网	12V 直流供电
	Venus lite (-T) 8-3ps	8[B]	CH0-7: ~3 ps	4	USB3.0/千兆网/万兆网	12V 直流供电
	Venus lite(-T) 4/5-2ps	4/5	CH0-3/4:~2 ps	2	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 产品的通道数和分辨率可定制；

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；

D. 所有出厂产品，官方提供一份实验室质量检验报告，参照《TR-01-03-010-1 Venus 设备质量检验报告-参考样本》。

4. 产品功能结构和应用场景

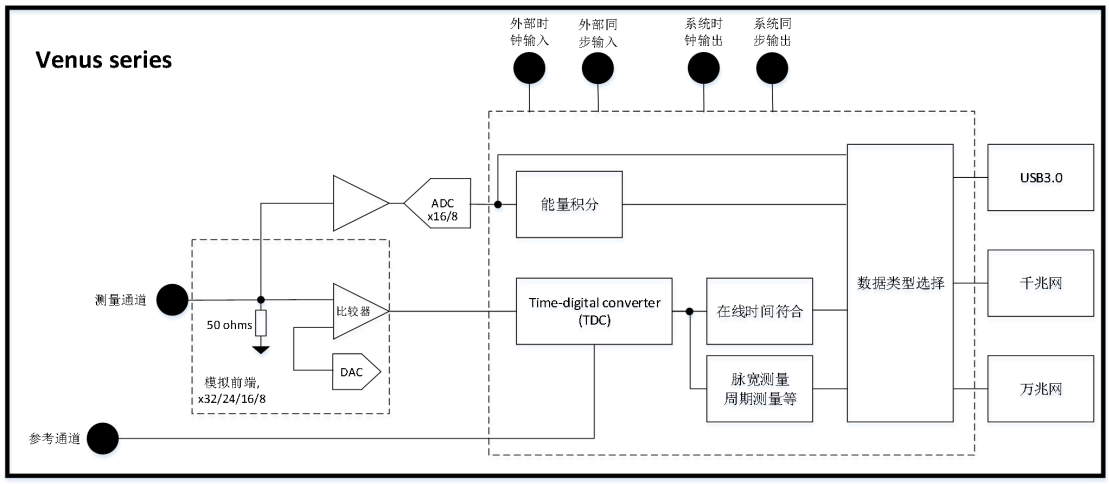


图 1 Venus 系列产品功能结构框图

Venus TDC 产品应用范围很广，几种典型的应用场景如下图所示。

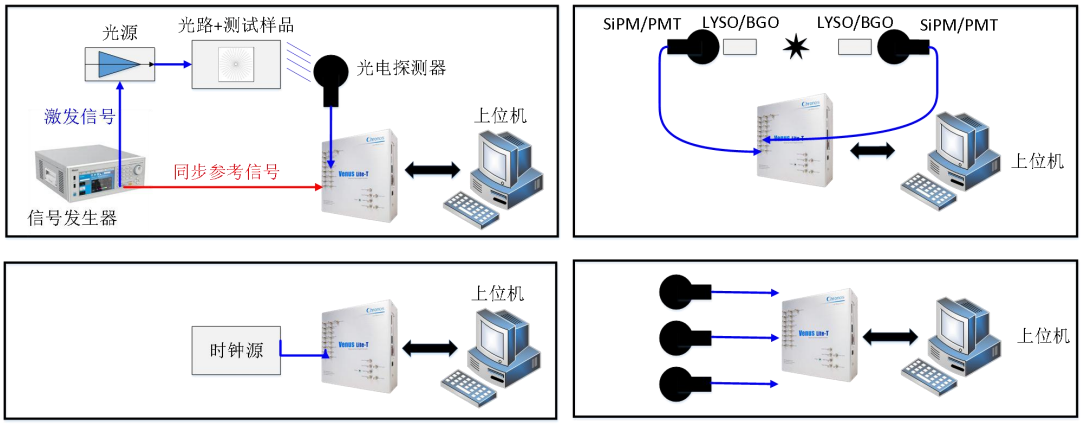


图 2 几种典型的应用实验场景

（左上：时间相关光子计数器：测试探测器信号与参考同步信号之间的时间差；

右上：符合探测器测试：测试符合探测器输出信号的能量和时间差；

左下：时钟源稳定性分析：测试时钟源时钟频率稳定性；

右下：宇宙线探测器阵列测试：测试随机宇宙线事件的到达时间并保存原始时间戳）

5. 产品外观尺寸

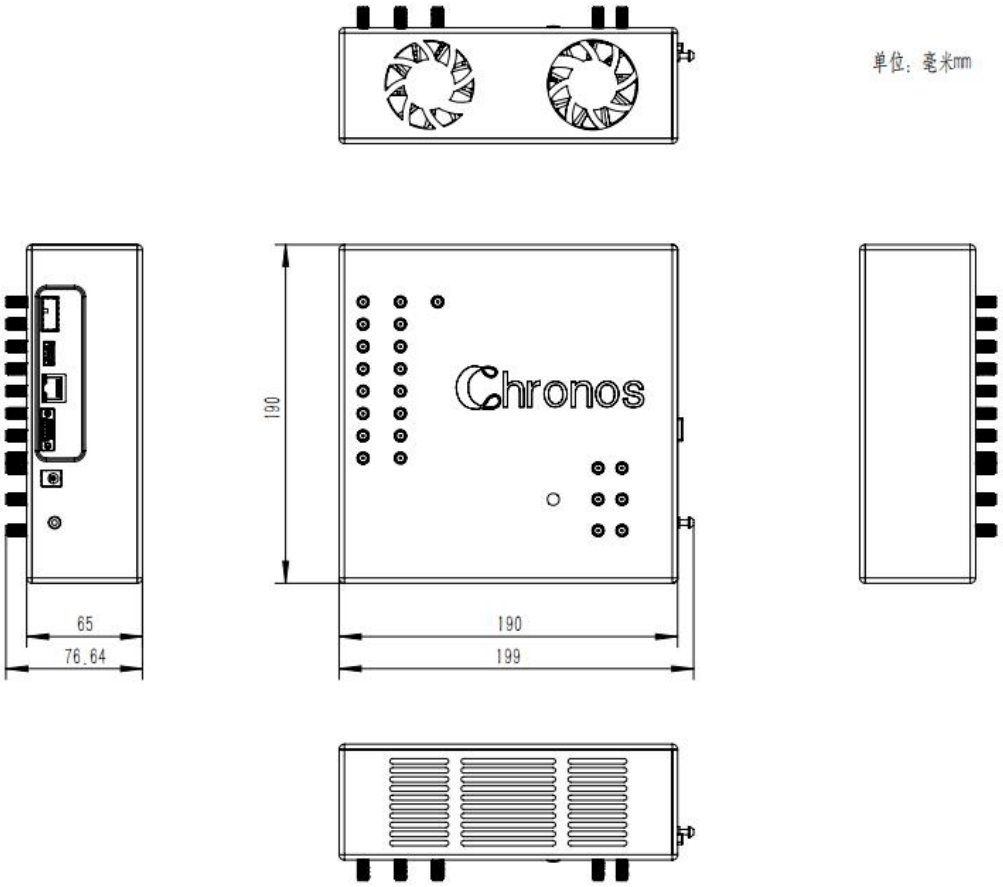


图 3 Venus Lite-T 系列产品外观和结构尺寸图

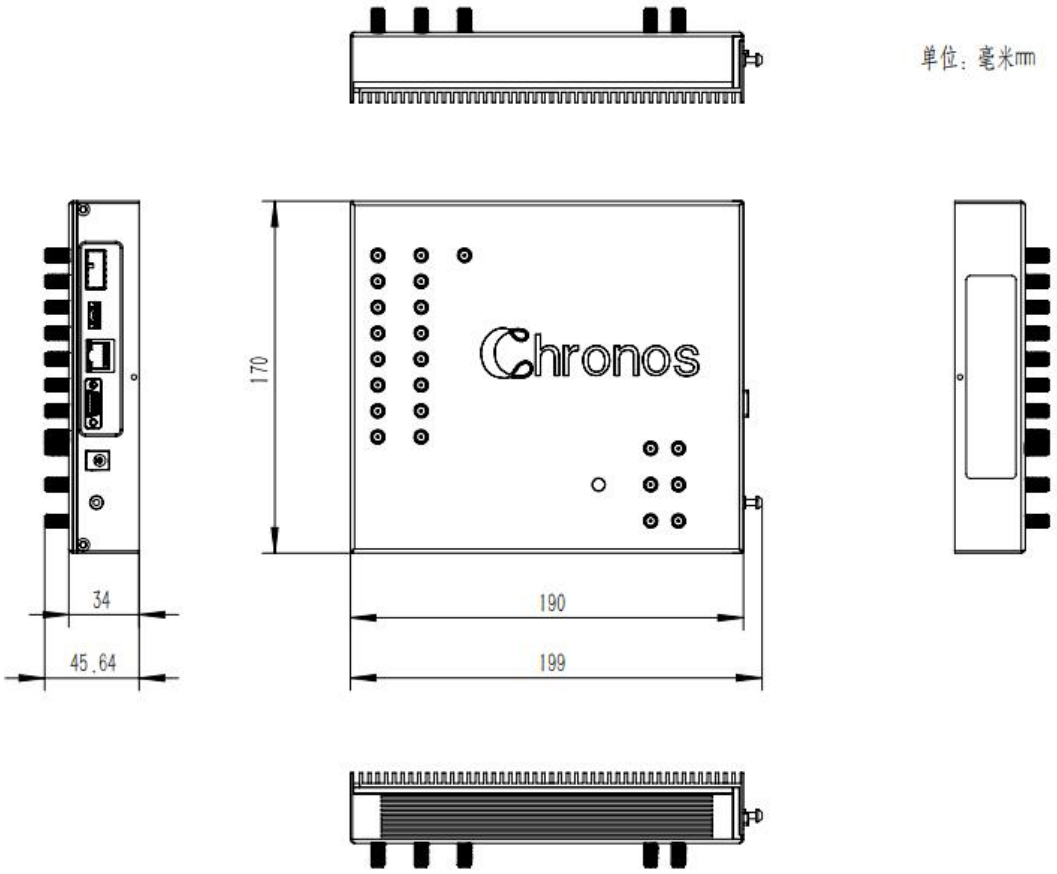


图 4 Venus Lite 系列产品外观和结构尺寸图

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		
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	-10 ℃ ~ 70 ℃	
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。

7.1 地环噪声的影响

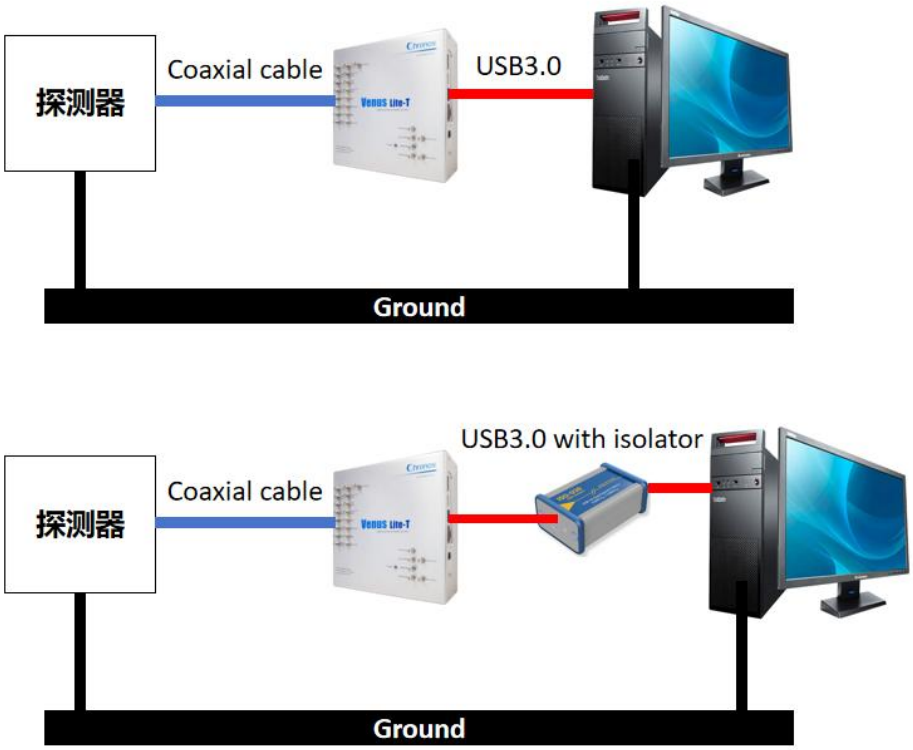


图 5 基于 USB3.0 接口的测试系统地环路噪声问题示意图

（上图：USB3.0 直连引入的地环路示意图；下图：接入 USB3.0 隔离器切断地环路示意图）

如果使用 USB 接口，将 Venus 与测试电脑（台式机或者服务器）连接，且 Venus 与探测器通过同轴电缆和 SMA 连接器连接。探测器一般都会与实验室地平面良好接地。由于 USB 线缆和同轴电缆会将测试电脑、Venus 和探测器的地

连接起来，与实验室地平面组成一个可能引入噪声的“地环”。或者，某些电脑的 USB 地与探测器的地会有电势差。如果遇到此问题或对测试噪声要求极高，建议：

（1）使用千/万兆网口进行测试；

（2）使用笔记本电脑进行测试；

（3）在测试电脑与 Venus 之间使用 USB3.0 isolator 进行电气隔离（磁耦合或者光耦合），切断地环影响。

由于 USB 3.0 超高速信号的特殊性，为确保最佳的信号完整性和隔离效果，我们为您推荐经过严格测试的外置 USB 3.0 隔离器（比如 7055-C/D，Intona）。该配件能有效切断地环路，保障您在复杂电磁环境下的测量精度和设备安全。

8. 上位机软件使用说明

8.1 软件安装说明

官方提供 Venus 上位机软件的安装包。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

若显示“用户账户控制”弹窗，选择“是”确认。

（2）选择程序安装路径

选择安装路径，建议不要安装到 C 盘（系统盘）。选择后，点击“下一步”。

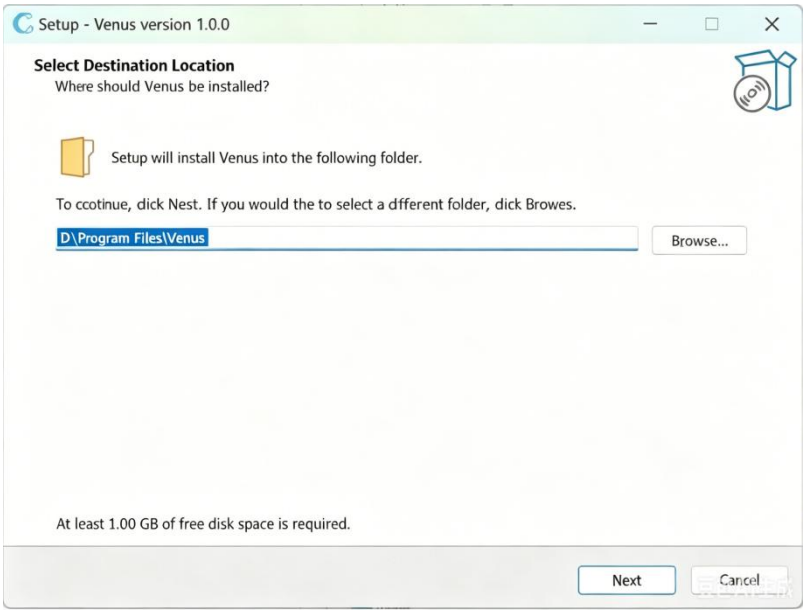


图 6 上位机安装步骤 2

（3）选择是否创建桌面快捷方式

点击“下一步”；

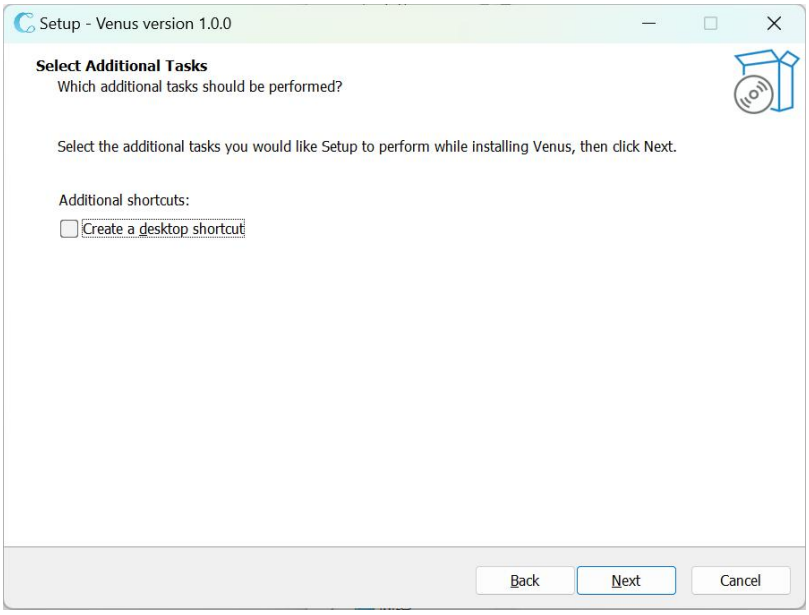


图 7 上位机安装步骤 3

（4）开始安装

点击“install”，将开始安装过程；

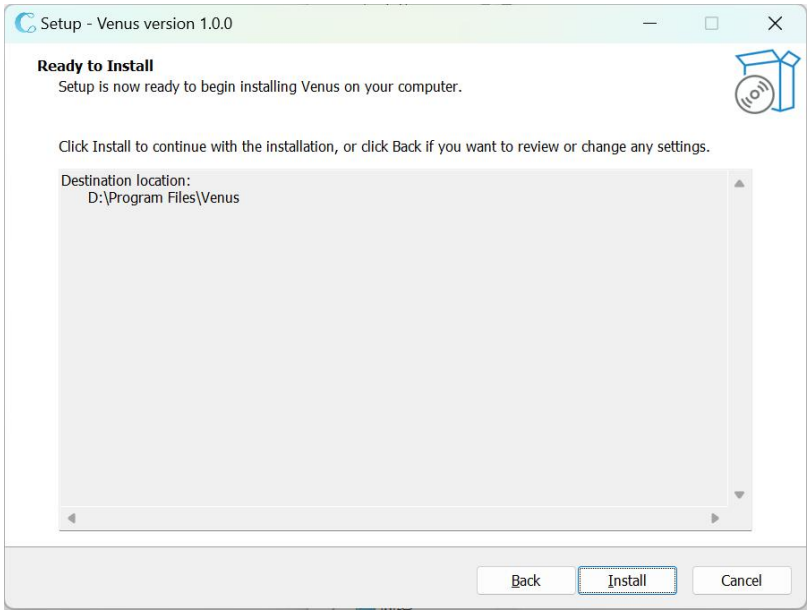


图 8 上位机安装步骤 4

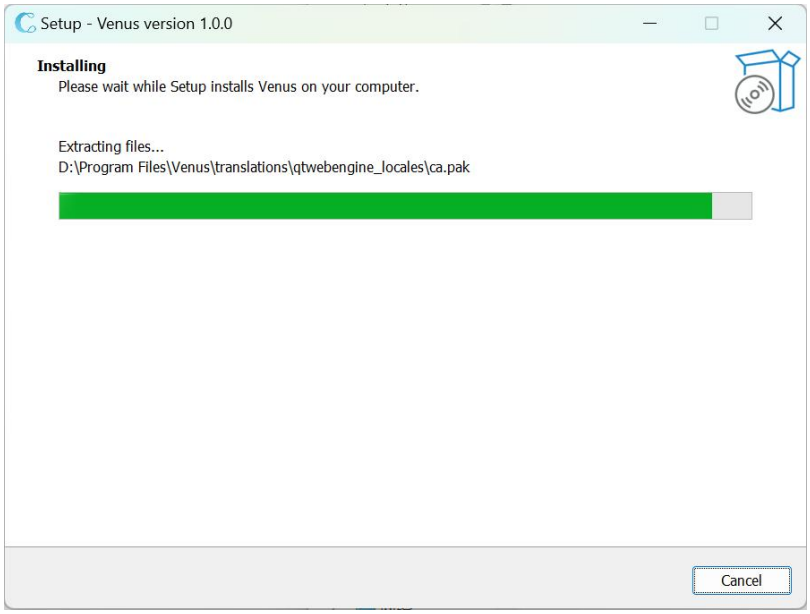


图 9 上位机安装步骤 4 安装过程

（5）安装成功

如果安装成功，将显示如下界面。

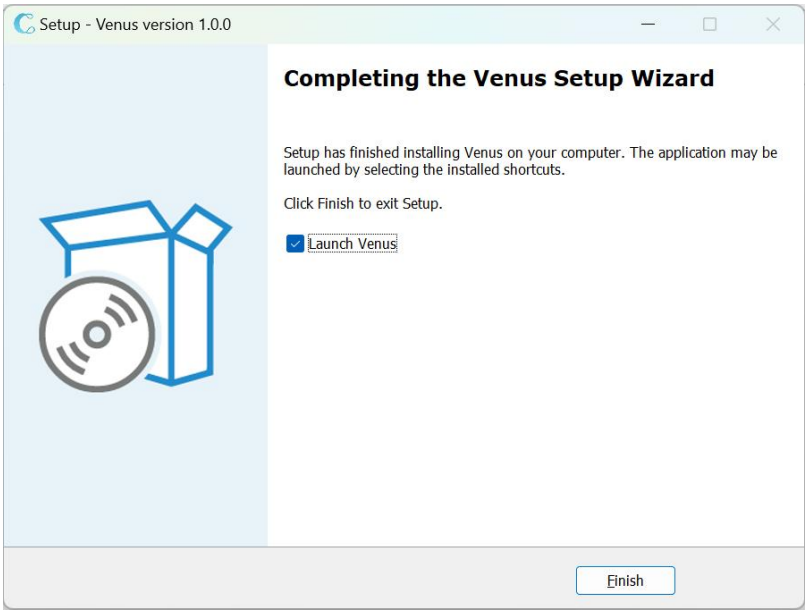

Setup - Venus version 1.0.0
Completing the Venus Setup Wizard
Setup has finished installing Venus on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Venus
Finish

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。

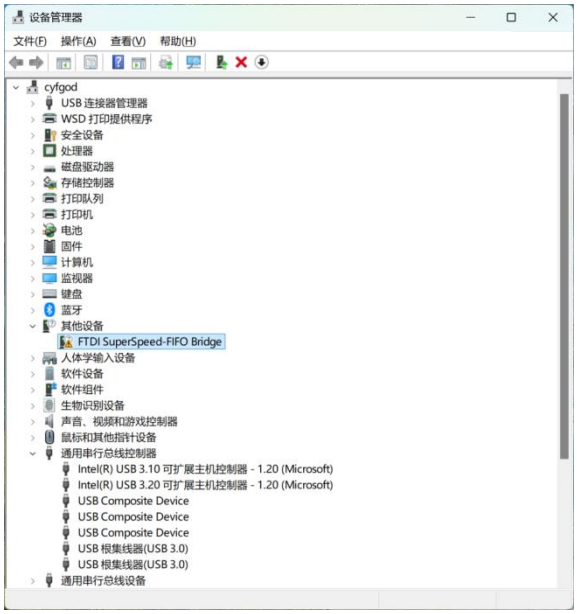


图 11 USB3.0 驱动设备管理器

（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

目前的 Linux 软件包主要以 Deb 包格式分发：

Deb 安装包文件：venus_X.X.X_jammy_amd64.deb (X.X.X 是软件版本号，jammy - 是 Ubuntu-22.04.05 LTS 版本的缩写)

（2） 打开终端，在安装包目录执行安装命令（以 V1.1.0 版本为例）：

->sudo apt install ./venus_1.1.0_jammy_amd64.deb

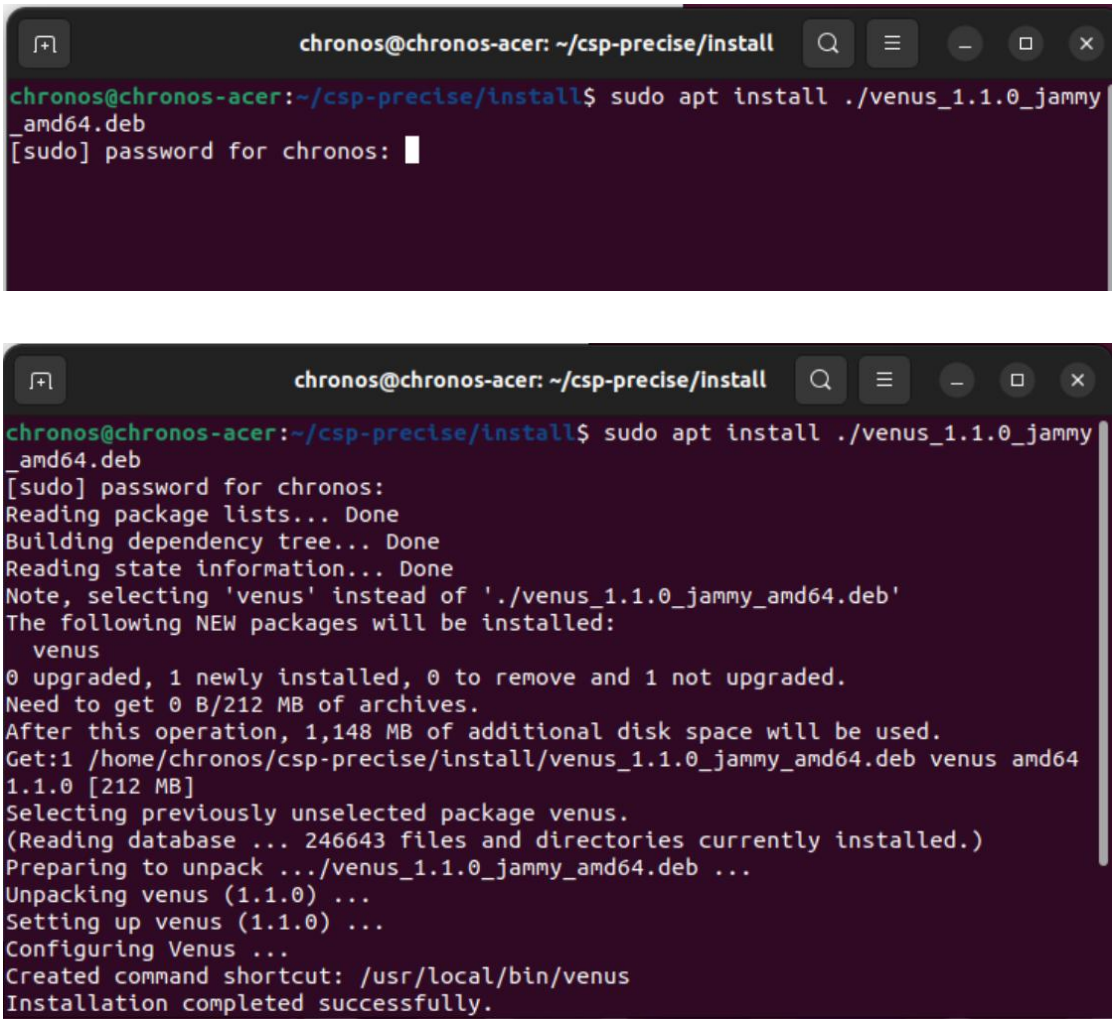


图 14 软件安装 Linux 命令

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，可以直接输入 venus 运行程序。

首先是会弹接口选择页面，选择设备接口。

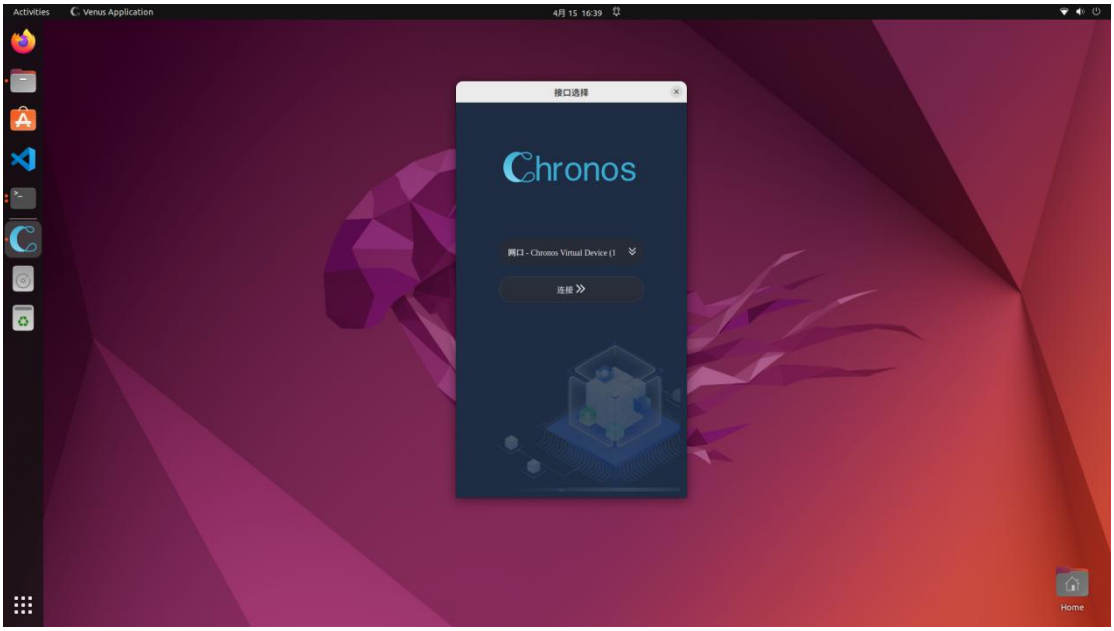


图 15 Linux 系统下的软件运行

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

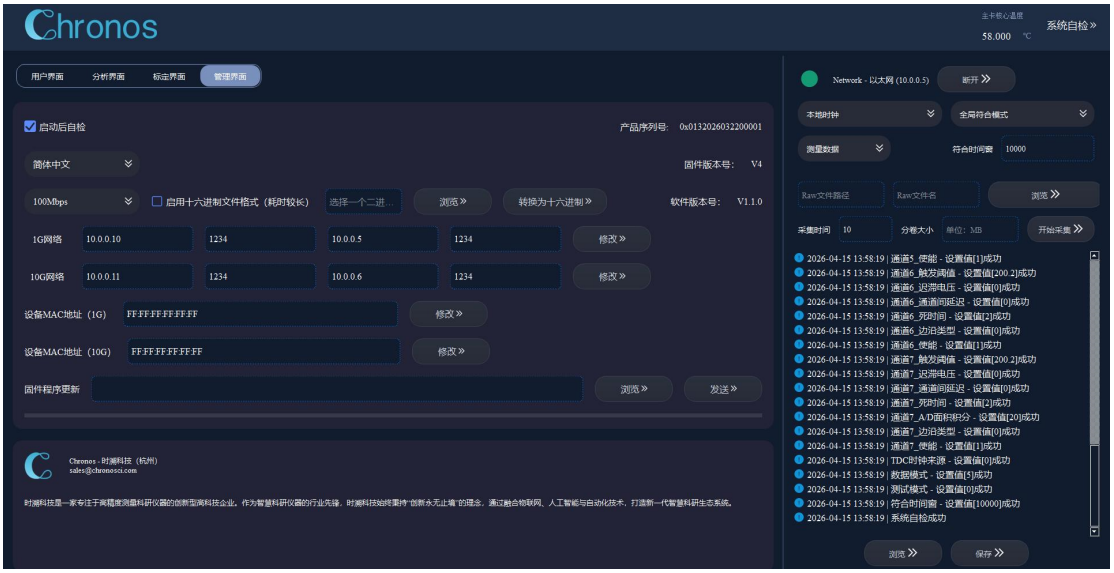




图 17 网口 IP 和端口地址修改页面

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（如 NVIDIA MAM1Q00A-QSA），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

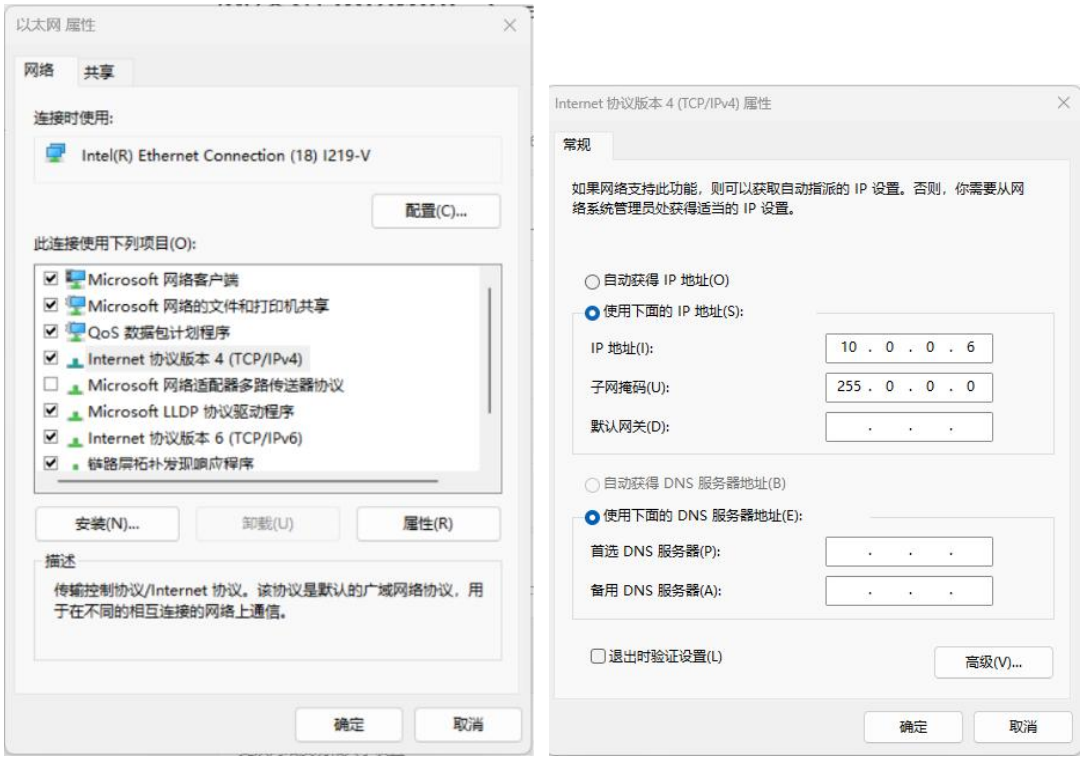


图 19 上位机电脑万兆网网络设置

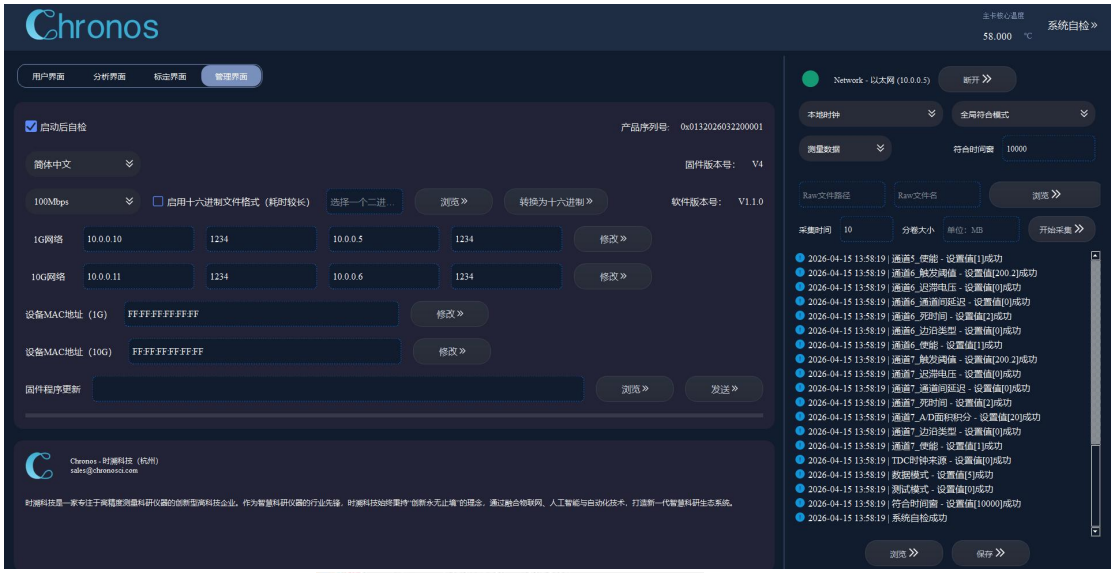




图 20 网口 IP 和端口地址修改页面

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） 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

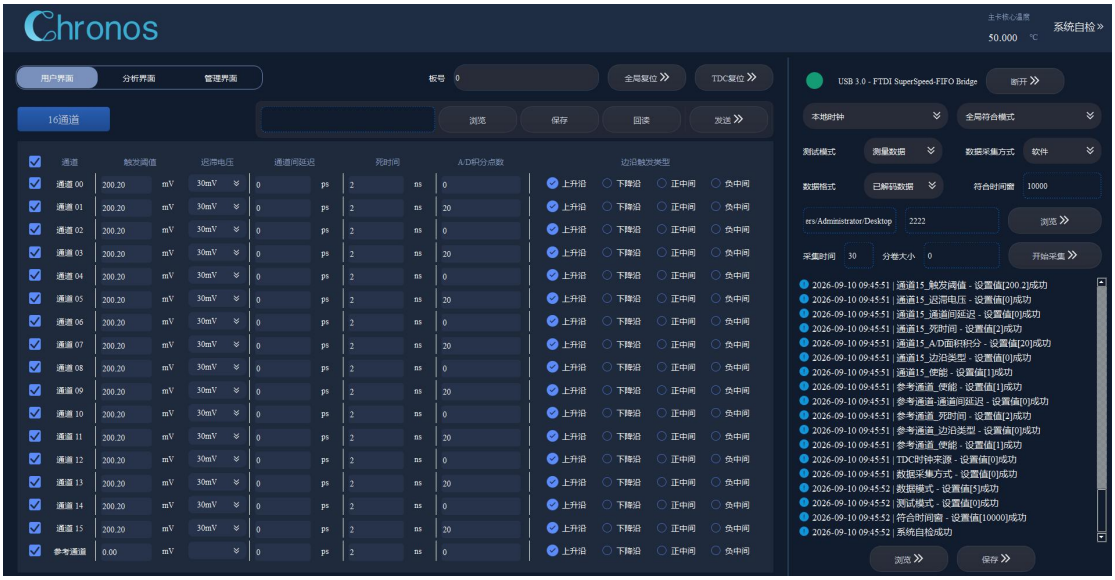


图 23 用户页面

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、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 积分点数、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、采集方式等参数；

（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 通道使能

每个通道配置页面前，都有使能勾选。勾选中后，本通道输入使能，不勾选，本通道输入关闭。用户可以选择目标测试通道，而关闭其他不使用的通道，来进行针对性测试和节约数据接口带宽，如下图所示。用户可选最上边选项框来全选或全不选来快速设置。



图 24 通道使能勾选

8.6.2 比较阈值配置

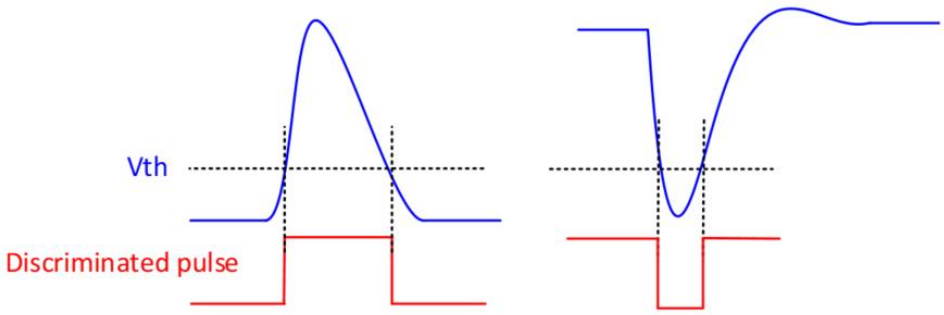


图 25 模拟信号比较功能示意图

Venus 模拟前端采用边沿定时（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），默认配置范围为-1500 mV～+1500 mV（如有需求可放开更大范围）。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）。

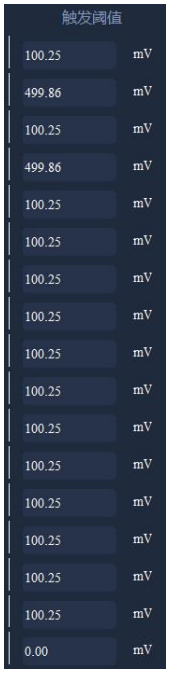


图 26 触发阈值配置 GUI 示意图

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。

8.6.3 各通道迟滞电压配置

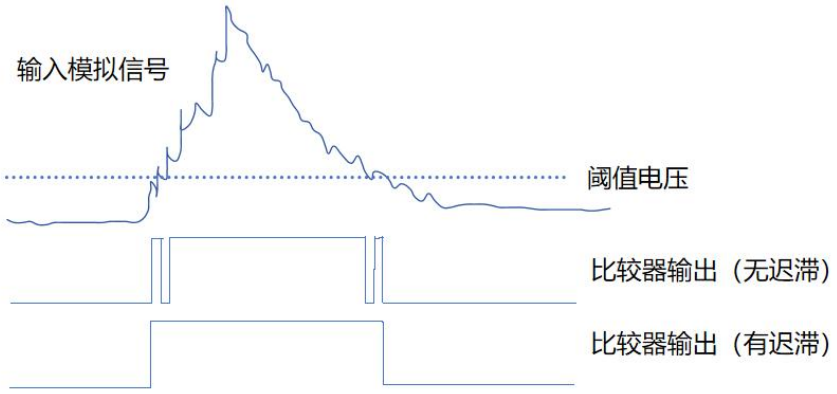


图 27 迟滞电压原理示意图

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，默认为 30 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。

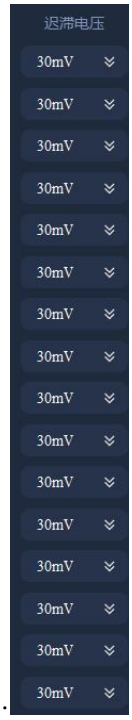


图 28 比较器迟滞电压配置 GUI 示意图

。

8.6.4 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 0.976 ps，调节范围为 1 us。

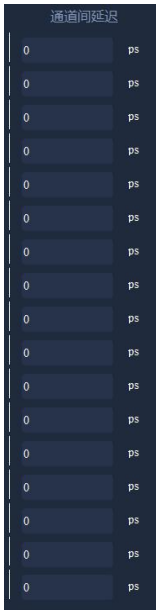


图 29 通道间延迟配置 GUI 示意图

注意，设备只支持正延迟配置，不支持负延迟配置。用户可以参考附录 J 进行快速设置。另外，为了方便用户离线生成对齐 延迟值，官方提供一个根据全局符合数据来进行延迟对齐查找表的 MATLAB 程序，让用户迅速制定延迟参数以对齐各个测量通道。

8.6.5 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）或反射（Reflection）现象的待测信号，设置死时间可有效滤除因振铃/反射产生的无效触发，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，在死时间内发生的重复触发不会再引入额外的死时间。

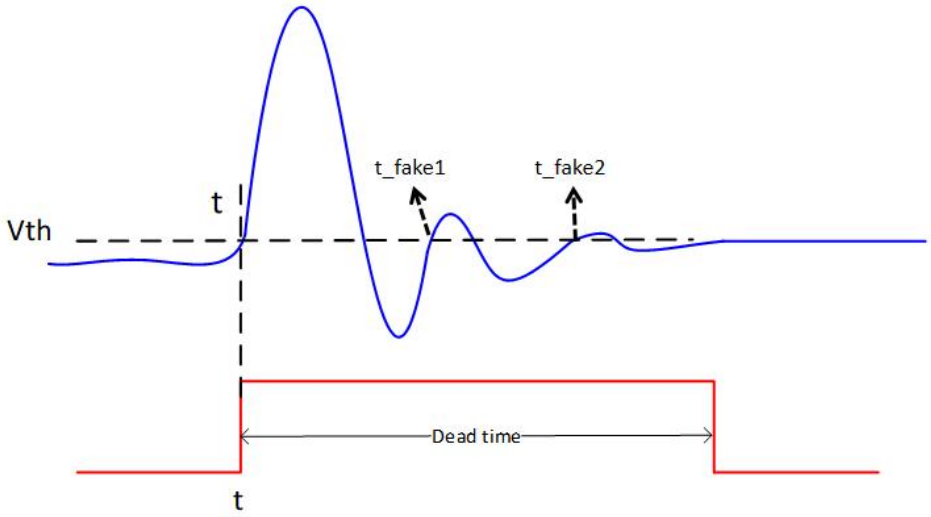


图 30 设置死时间以消除“振铃”带来的无用数据读出

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。

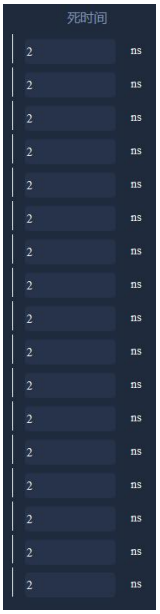


图 31 通道时间测量死时间配置 GUI 示意图

8.6.6 ADC 积分点数

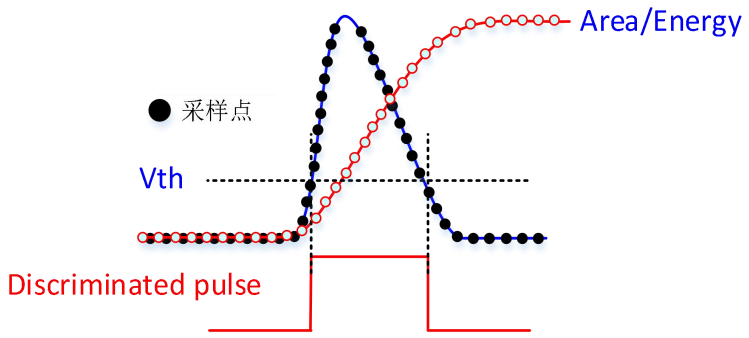


图 32 能谱测量数据模式逻辑框图

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。

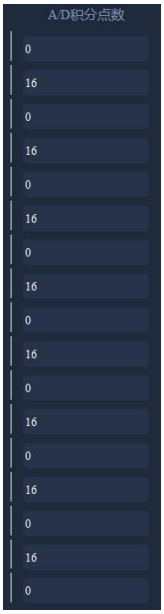


图 33 A/D 通道采样点个数配置 GUI 示意图

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，用户无需担心输入信号基线的变化对能量测量的影响。

8.6.7 通道参数快速设置（One-Key）

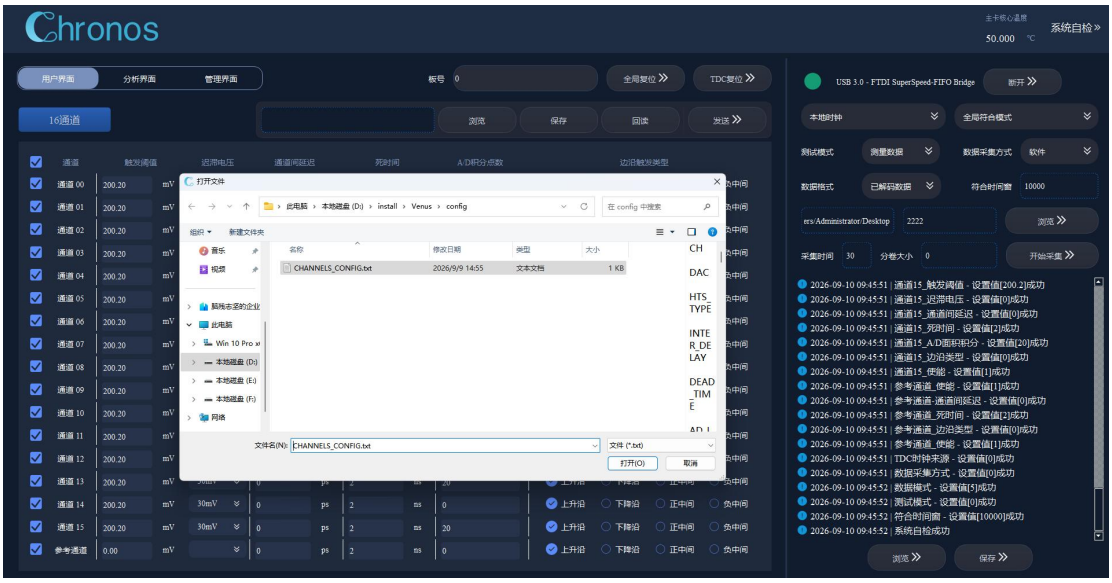


图 34 One-key 配置方式

用户可通过上位机软件界面或导入配置文件的方式，配置各通道的触发阈值、延迟值、死时间值和 ADC 积分点数。通过文件配置用户可以一键配置所有参数（One-Key）。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 CHANNELS_CONFIG.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参数配置；

（3）文件格式说明： CHANNELS_CONFIG.txt 文件格式如下所示。文件左侧列为通道号，右侧列为对应通道参数值。

CHANNELS_CONFIG.txt

	CH	DAC	INTER_DELAY	DEAD_TIME	AD_INTEGRATION_POINT
1	0	19.80	0	2	0
2	1	40.10	20	4	20
3	2	59.82	40	8	0
4	3	80.13	60	12	40
5	4	99.85	80	16	0
6	5	120.15	100	20	60
7	6	139.88	120	24	0
8	7	160.18	140	28	80
9	8	179.90	160	32	0
10	9	200.20	180	36	100
11	10	250.09	200	40	0
12	11	299.98	250	44	120
13	12	349.86	300	48	0
14	13	399.75	400	52	160
15	14	500.10	500	56	0
16	15	800.00	600	60	200
17					

图 35 CHANNELS_CONFIG.txt 文件格式要求

（左侧列：通道号；右侧列：触发阈值（单位：mV）、延迟值（单位：ps）、死时间值（单位：ns）、ADC 积分点数）



图 36 CHANNELS_CONFIG.txt 文件保存当前配置

另外，用户可以通过软件自动保存当前页面的所有通道配置参数，更新 CHANNELS_CONFIG.txt 文件。这个功能方便用户重复配置多次相同的试验参数。

8.6.7 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： 若需测量脉冲下降沿（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。

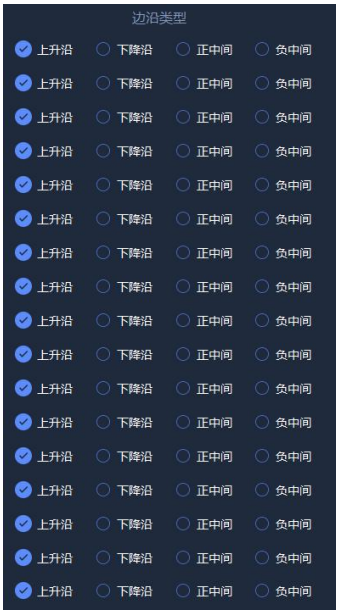


图 37 边沿采集模式配置 GUI 示意图

配置示意图如下所示。

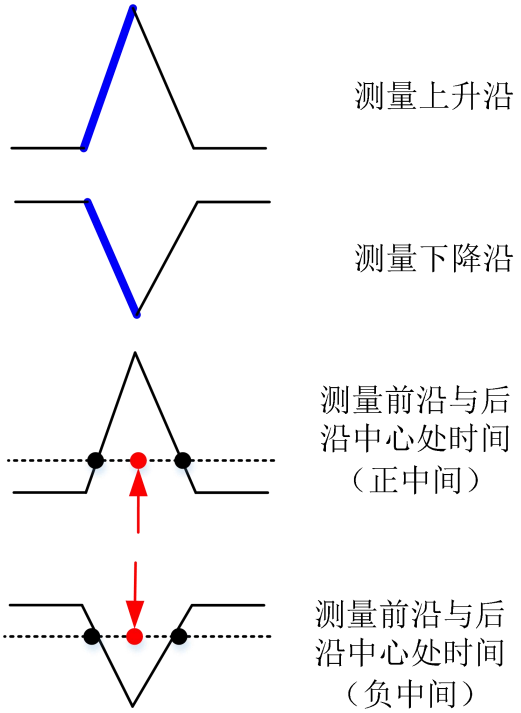


图 38 测量边沿时间戳信息选择举例（假如测量第一个边沿时间戳信息，如果输入正向脉冲，用户应选择测量上升沿；如果输入负向脉冲，用户应该选择测量下降沿。如果信号 Time Walk 或者基线漂移较大，且信号两个边沿定时性能均佳时，可以选择中间模式。）

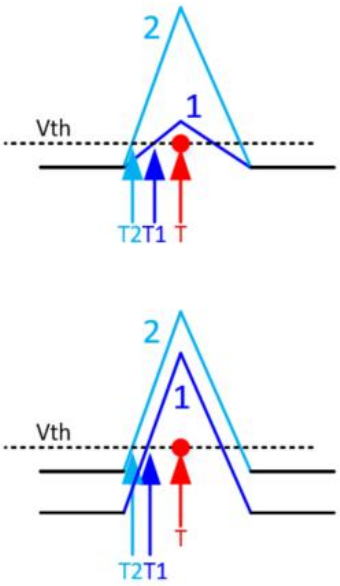


图 39 MID 测量方式举例说明（以正中间模式为例。上图：输入信号 1 幅度较小，过阈时间 T1，输入信号 2 幅度较大，过阈时间 T2。T1 与 T2 之间存在 Time Walk。采用正中间方式，对于某些信号形状固定的情形下，能够有效减小 Time Walk 的影响；下图，输入信号 1 直流基线较小，过阈时间 T1，输入信号 2 直流基线较大，过阈时间 T2。T1 与 T2 之间存在定时差。采用正中间方式，对于某些信号形状固定的情形下，能够有效减小定时差的影响。）

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。

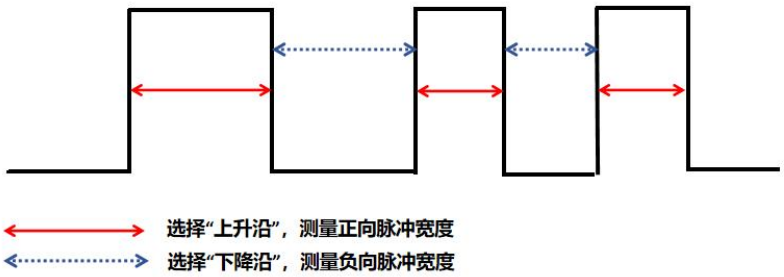


图 40 测量边沿时间戳信息选择举例（选择“上升沿”，TOT 测量中测量正向脉冲宽度；选择“下降沿”，TOT 测量中测量负向脉冲宽度。）

8.6.8 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 内部时钟芯片产生，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。在 V1.3 版本硬件，仅支持 25 MHz 时钟输入；在 V1.4+版本硬件，支持 10 MHz 或 25 MHz 两种外部时钟输入。

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. 选择采集启动方式

Venus 支持软件启动采集（软件方式，常规），或者使用外部脉冲信号（硬件方式）开始采集，如下图所示。软件/硬件启动采集方式选择可通过上位机软件配置。



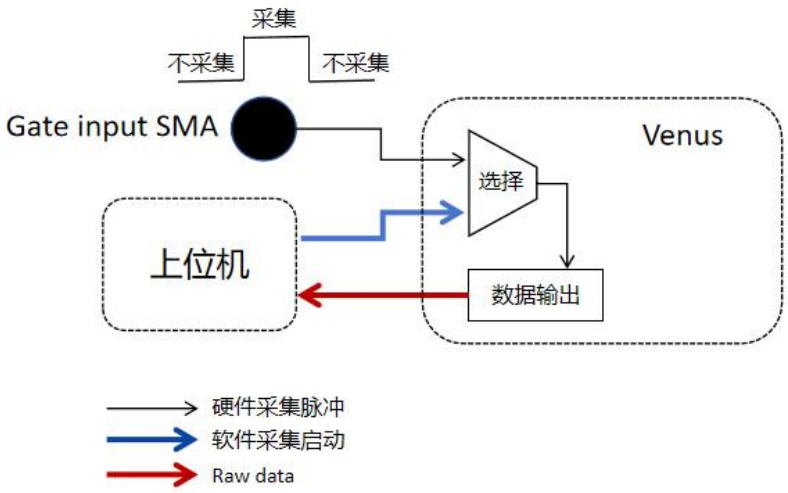


图 41 两种采集启动方式

- 软件采集方式：一般用户建议使用的方式。用户只需要通过上位机软件

或 API 设置采集路径和采集时间，然后通过软件开始采集即可。下文所有的数据采集介绍都基于此方式；

- 硬件采集方式：通过设备上 Gate input SMA 输入脉冲，设备会在脉冲高电平时段输出 Raw 数据，在低电平时段关闭 Raw 数据输出。用户可以先通过 API 启动数据采集监听，然后输入采集脉冲，设备会保存在采集脉冲高电平时段所有产生的 Raw 数据，试验完成后用户需要关闭采集监听。硬件采集可以用在以下特殊情形：
  - 高精度采集时长控制：相比软件控制，硬件采集脉冲可以保证时长很精确，精度±10 ns，在某些对采集时长有严格要求的情形下非常有用；
  - 多设备严格同时采集：在某些情形下，需要对多台设备进行高精度同时采集，通过硬件脉冲可以很精准的进行同步采集，如下图所示。

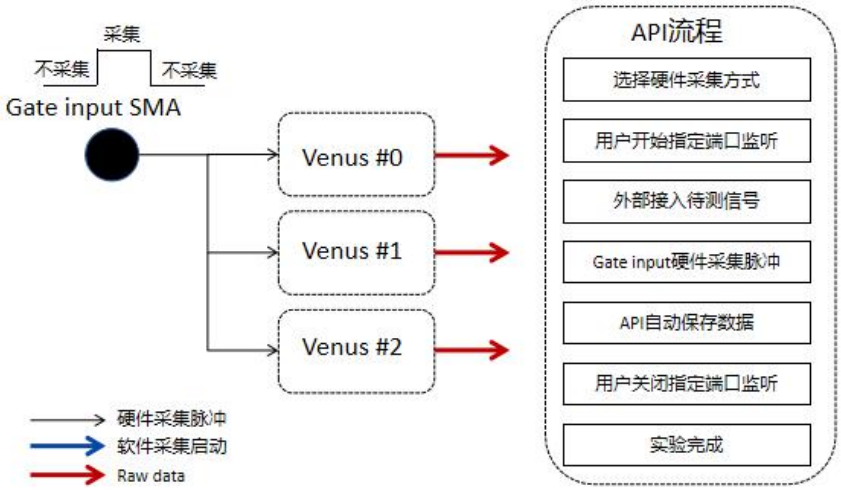


图 42 使用外部硬件采集脉冲实现多设备高精度同时采集

4. 选择采集模式

根据测量需求，从以下模式中选择其一：

- 时间戳模式
- 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
- 带时间戳的全局符合模式
- 单沿时间数据模式
- 全局符合数据模式 2

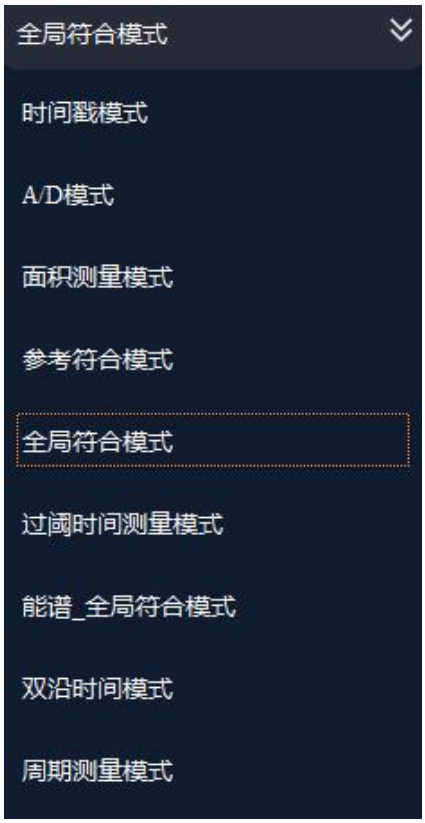


图 43 数据采集模式选择

数据采集模式解释如下：

（1） 时间戳模式（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 采集所有通道产生的时间戳，然后输出触发类型指定边沿的时间戳。这个模式会输出两类时间戳，一类叫做时区时间戳（Time zone），另一类叫做事件时间戳（Events time stamp）。时区时间戳全局有效，表示较粗和较大的时间范围，事件时间戳表征着每个通道的触发较细时间戳信息。综合两个时间信息可以还原出事件发生的完整时间戳。具体数据格式参照 12.1 节；

（2） 双沿时间戳数据模式（所有通道有效）：类似于（1），此模式也是输出原始的触发时间戳信息。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。在高计数率情况下，为了提高数据传输效率，建议用（1）时间戳模式。如果普通应用情况下，为了方便数据分析，建议使用本模式；

（3） 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（包含两个符合通道的通道号、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；此格式将所有通道的双通道时间差输出，关于进一步的 N 重符合数据筛选和处理，参见 8.7.11 节。

在线符合逻辑图参见附录 F。

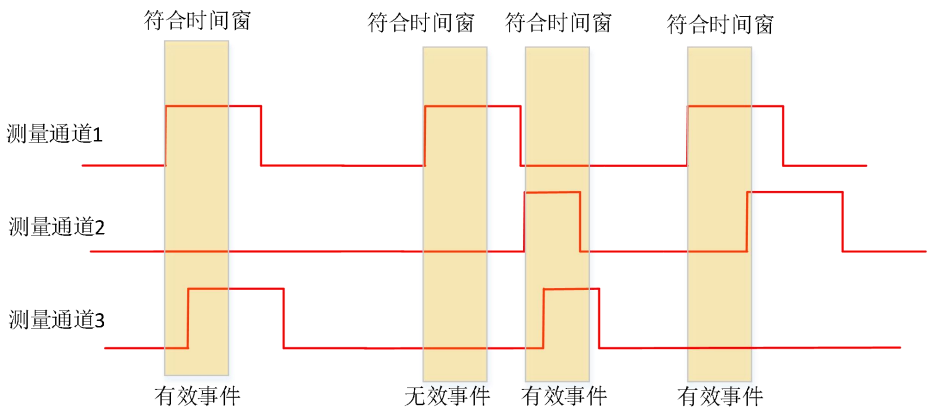


图 44 在线时间全局符合逻辑（三通道举例，实际为全通道参与符合算法）框图（测量事件边沿：上升沿）。用户设置符合时间窗，选择时间全局符合数据模式，系统会根据两个事件的时间差信息，判断是否满足符合条件。

注意的是，全局符合数据模式是输出**所有**双重符合的物理事件。举例，如果通道 CH0、CH1、CH5、CH8 四个通道发生了符合事件，那么 Venus 将输出以下符合数据（比如本次事件按照以下时间先后发生: CH1-> CH5 -> CH0 -> CH8）：

Coincidence Raw data1: {CH1, CH5, Delt_T(CH1,CH5)};

Coincidence Raw data2:{CH1, CH0, Delt_T(CH1,CH0)};

Coincidence Raw data3:{CH1, CH8, Delt_T(CH1,CH8)};

Coincidence Raw data4:{CH5, CH0, Delt_T(CH5,CH0)};

Coincidence Raw data5:{CH5, CH8, Delt_T(CH5,CH8)};

Coincidence Raw data6:{CH0, CH8, Delt_T(CH0,CH8)};

因此，N 个通道发生符合，Venus 将输出$\frac{N(N-1)}{2}$个双重符合数据对，即所有两个通道时间差在时间窗内时都输出，不会造成符合数据损失。

Venus 上位机软件和 API 包括 N 重符合分析模块，用户可以直接调用进行更高级的 N 重符合计数率和时间差分析，参见 8.7.11 节。

（4） 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；

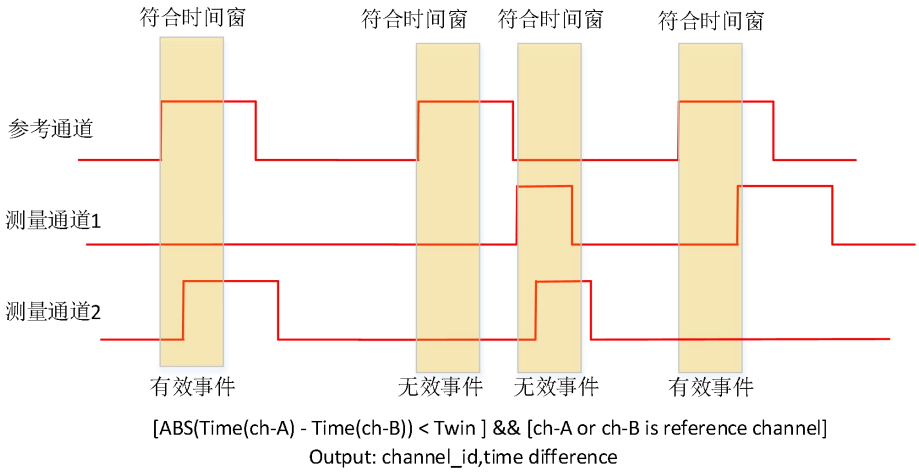


图 45 在线时间参考符合逻辑（参考通道+两个测量通道举例，实际为全通道参与符合算法）

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。

（5） 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；

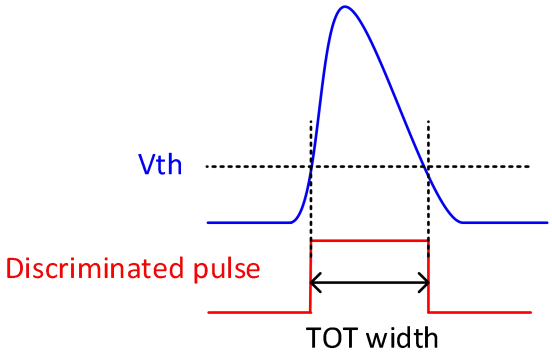


图 46 过阈时间测量逻辑框图（以正信号为例，用户应该“上升沿”边沿采集模式）

（6） 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；

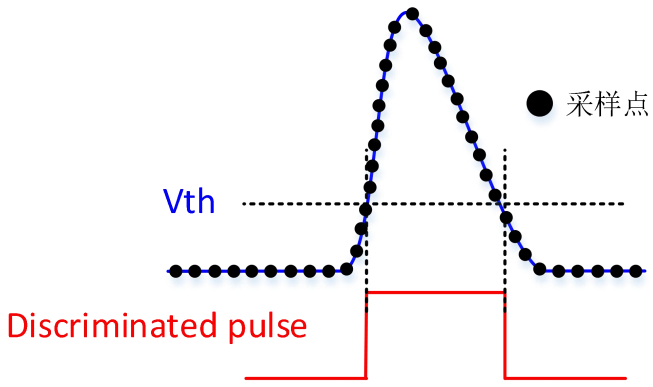


图 47 ADC 测量数据模式逻辑框图

（7） 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；

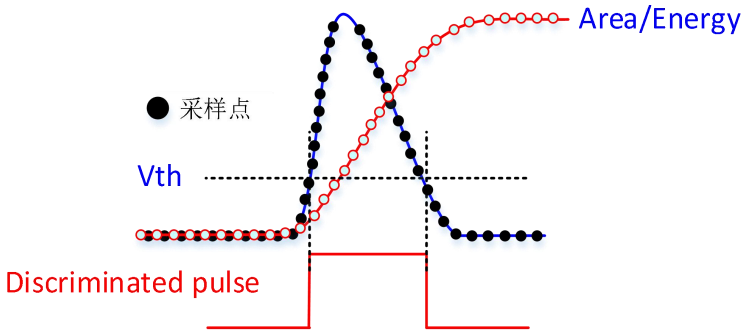
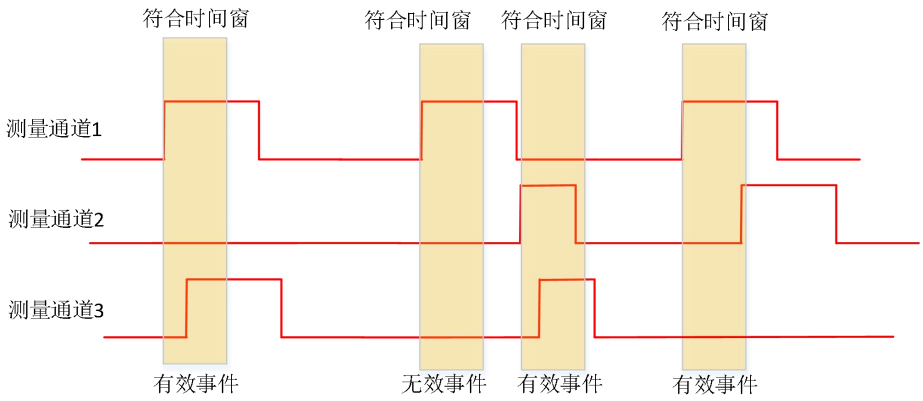


图 48 能谱测量数据模式逻辑框图

（8） 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 49 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。本模式与全局符合数据基本相同，只是增加了两个符合通道信号的能量信息。

（9） 周期测量数据模式（通道 0-通道 3 有效，但是每次只能测量一个通道）：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；

（10） 带时间戳的全局符合数据模式（所有通道有效）：在此模式下，符合逻辑与全局符合模式完全一样，不同的是最终输出的不是两个通道的时间差，而是第一个事件发生的时间戳信息。具体数据格式参照 12.9 节；

（11） 单沿时间戳数据模式（所有通道有效）：此模式也是输出原始的触发时间戳信息。在此模式下，当通道过阈值触发后，Venus 会将信号的上升沿或者下降沿的时间戳信息（用户选择边沿触发类型）输出。用户可据此分析单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 12.10 节。在高计数率情况下，为了提高数据传输效率，建议用（1）时间戳模式。如果普通应用情况下，为了方便数据分析，建议使用本模式；

（12） 全局符合数据模式 2（所有通道有效）：与全局符合模式原理相同。只是对符合事件的时间戳流进行更精细序列标记。具体数据格式参照 12.11 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
时间戳	Timestamp	所有通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源等
双沿时间戳	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
时间参考符合	Ref-coins	参考通道+所有测量通	时间甄别阈值/死时间/

		道	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
周期测量	Period	通道 0-3，且每次只能测试 1 个通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源等
带时间戳的全局符合数据模式	Global-coins with timestamp	所有通道	时间甄别阈值/死时间/时间延迟/TDC 时钟来源等
时间全局符合 2	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

8.6.9 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。



图 50 测量数据保存配置



图 51 数据保存格式

数据保存格式分为二进制、十六进制和解码格式。

（1）二进制：默认保存为.bin 格式，用户需要自行解码，存储效率高；

（2）十六进制：默认保存为.dat 格式，用户需要自行解码，存储效率中；

（3）解码格式：默认保存为.csv 格式，数据已经转换。比如时间转换为了 ps，ADC 码值转换为了 LSB，用户可以直接使用，无需再解码。

二进制和十六进制解码可以通过第 12 章的数据格式进行解码，可以参考 9.6/9.7 节的参考离线分析脚本进行解码。一般应用中，解码格式是比较简单的使用方式。特别注意的是，对于万兆网接口，为了提高存储性能，一律使用二进制存储。

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
通道使能	使能/不使能	CH0-CH15	不使能的通道不参与

			数据传输和在线符合
DAC	-1500 mV ~ +1500 mV	CH0-CH15	甄别阈值（软件限制设置范围，如需要更大范围可开放）
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

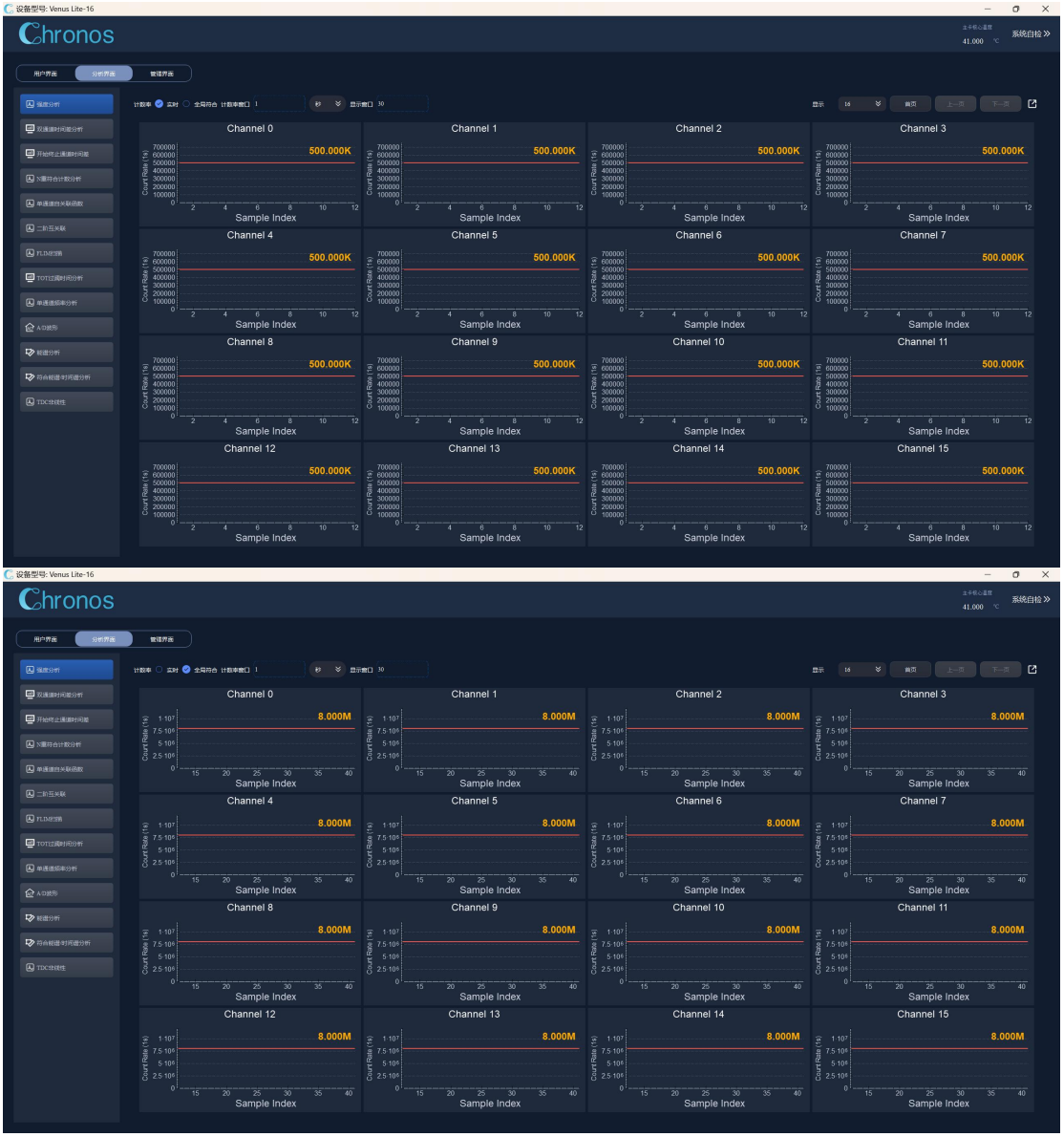


图 52 各通道强度（计数率）分析（实时计数率和符合计数率）

如上图所示，用户可进入“强度分析”功能页，按用户设置的计数率窗口（范围：10 ms–10 s）周期刷新，显示所有通道的实时计数率（Singles Count Rate）和符合计数率（Coincidence Count Rate）。用户还可以设置显示窗口，观察指定范围内的计数率曲线（如果设置为 0，则显示全时段历史曲线）。该显示窗口可独立弹出以随时实施观测。此功能为系统调试和性能评估提供了关键工具。

注意，此处的符合计数率包括所有双重符合事件的计数，具体见 8.6.8 节描述。举例，如果 N 个通道同时输入 1 MHz 的同步脉冲，所有通道都发生了符合

事件，那么由于任意通道都与其他所有通道发生符合，那么每个通道的符合计数率为(N-1) MHz。

8.7.2 双通道时间差分布直方图

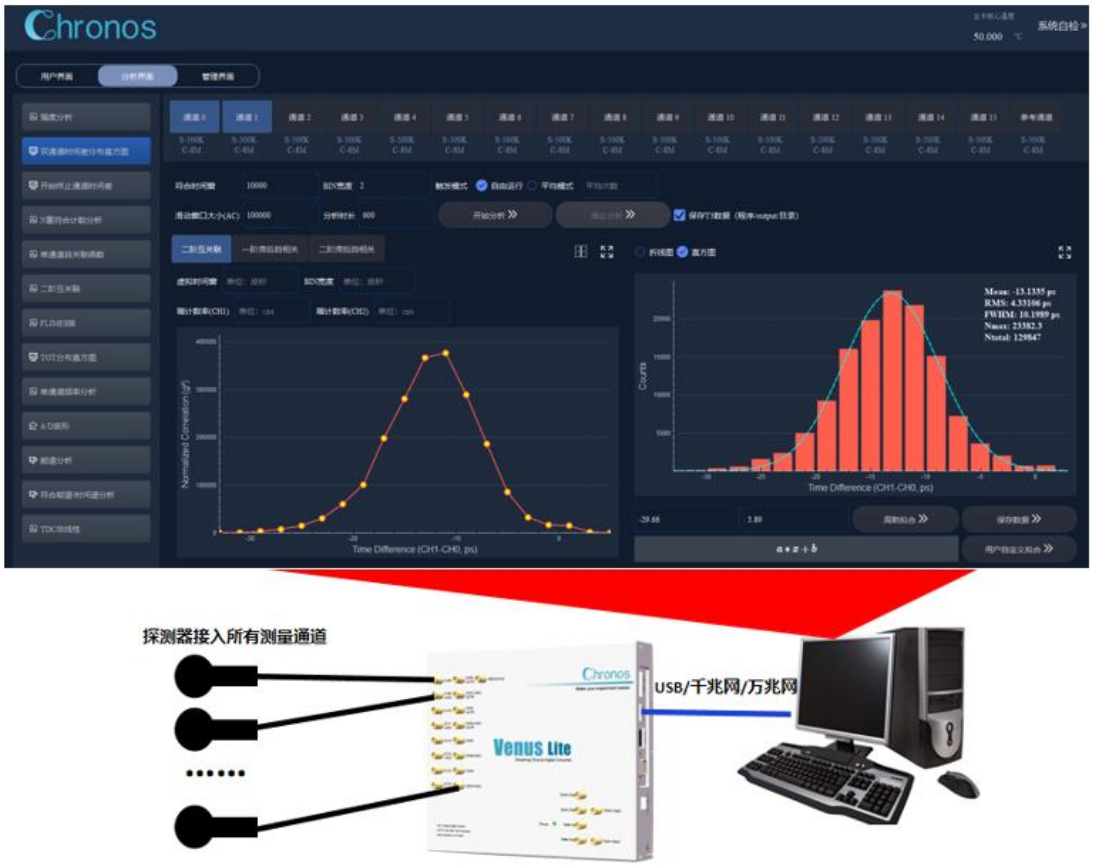


图 53 双通道时间差分布直方图

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（自由运行/平均）及平均次数、直方图显示 BIN 宽、BIN 数目等参数，随后点击“开始采集”。

- 采集时间：采集数据持续时间，单位 s；
- 符合时间窗：全局符合时间窗，两个通道信号到达时间在时间窗之内时作为一次符合事件；
- 触发模式：自由运行是实时采集并显示每个符合事件，平均模式是按照平均次数对输入信号的时间差进行取平均处理；
- BIN 宽：设置直方图显示 BIN 宽度；
- 双通道时间差图可选直方图或折线图，折线图画图效率更高，直方图画图效

率略差，可能会出现卡顿现象。如果用户必须使用直方图形式观察结果，请按照下述方法进行设置：

- 当符合时间窗设置较大时，为了减小软件画图压力，BIN 宽需设置大一些。当时间窗设置较小时，BIN 宽可以设置较小。举例说明，建议用户按照以下步骤进行设置：
  - 设置符合时间窗较大（超过两个通道的延迟时间），比如 100 ns；
  - 设置 BIN 宽度 20 ps；
  - 开始采集，并画时间差分布直方图；
  - 找到目标分析的时间差分布直方图的峰值，然后拟合或估计两个通道的时间差平均值。比如得到 Ch1-Ch0 平均值为 50 ns，指的是 Ch1 比 Ch0 平均晚 50 ns；
  - 然后在用户界面中的通道延迟值中设置，将 Ch0 延迟设置为 50 ns；
  - 然后将符合时间窗再设置较小，比如 1 ns，并且将 BIN 宽设置为 1 ps；
  - 然后开始采集，Ch0 和 Ch1 两个通道的时间差平均值会在 0 ps 附近，这样用户可以进行进一步的分析并高效快速实时画图。

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。采集完成后，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）等。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 54 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。



图 55 双通道互关联系数实时计算

Venus 还提供实时的时间差一阶和二阶归一化滞后自相关系数结果、双通道二阶互关联系数（g^2_{AB}）计算结果。其中，滞后自相关系数计算公式参照附录 G，

g^2_{AB}的计算公式参考附录 I。在计算g^2_{AB}时，用户需要输入两个通道的暗计数值。用户可以先评估探测器的暗计数值，如果用户忽略暗计数值，不输入即可，软件计算时则默认暗计数值为 0。

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。



	A	B	C	D
1	index	channel_a	channel_b	timediff_ps(Tb-Ta)
2	50014	0	1	-12
3	50015	0	1	-4
4	50016	0	1	-6
5	50017	0	1	-6
6	50018	0	1	-6
7	50019	0	1	-9
8	50020	0	1	-10
9	50021	0	1	-6
10	50022	0	1	-6
11	50023	0	1	-6
12	50024	0	1	-22
13	50025	0	1	-13
14	50026	0	1	-8
15	50027	0	1	-13
16	50028	0	1	-13
17	50029	0	1	-12
18	50030	0	1	-12
19	50031	0	1	-12
20	50032	0	1	-16
21	50033	0	1	-12
22	50034	0	1	-12
23	50035	0	1	-19
24	50036	0	1	-14
25	50037	0	1	-16

图 56 双通道时间差分析过程中 T3 数据的保存

此外，Venus 还支持采集过程中 T3 时间差数据的实时保存，用户只需要勾选“保存 T3 数据”即可。输出的数据（csv 格式）将保存在程序安装目录下的 output 文件夹中。

8.7.3 Start-Stop 分组测试

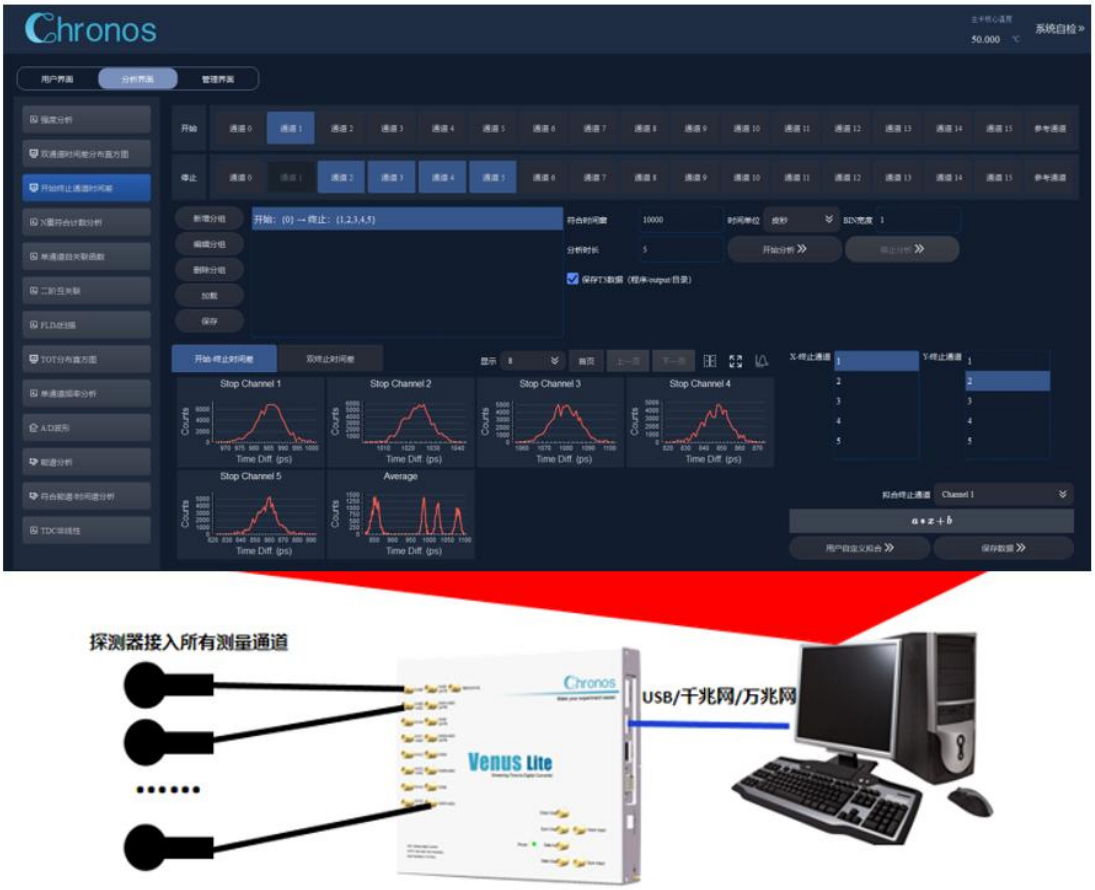


图 57 Start-stop 分组测试界面

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。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。

本分析功能常用于时间相关单光子计数器（Time-Correlated Single Photon Counting，简称 TCSPC）、双光子符合、多通道关联分析等场景。下面根据一个测试案例进行详细说明。

Start-stop 原理说明示意图如下所示。以测量上升沿为例。

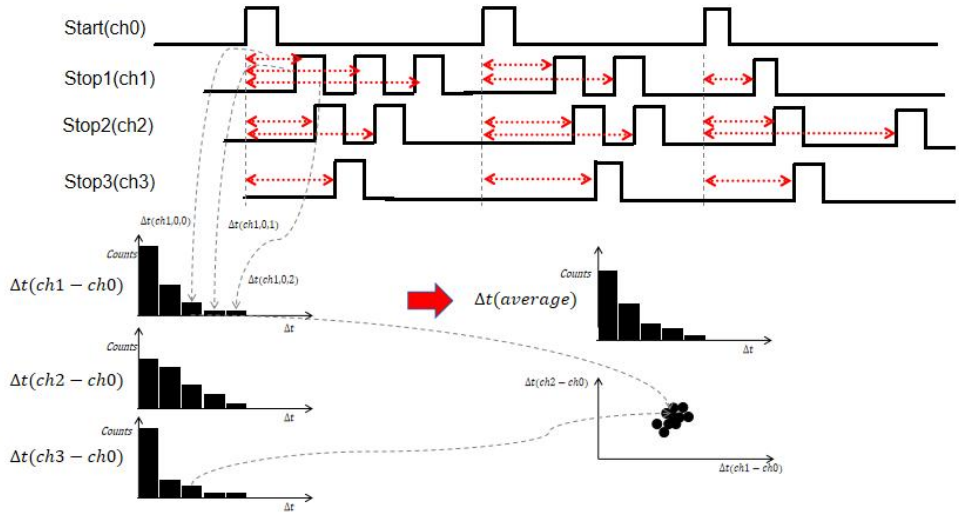


图 58 Start-stop 统计示意图（以上升沿触发为例，以 CH0 作为 start 通道，CH1/2/3 为 Stop 通道为例。实际用户可以按照实验要求自行分组）

（1）首先，用户选择 Start 通道，本例为通道 0;

（2）然后，用户选择任意个 Stop 通道，本例为通道 1/2/3，通道 0 和通道 1/2/3 组成了一个测试分组;

（3）然后，用户设置符合时间窗，符合时间窗也为时间差统计直方图横坐标的最大范围;

（4）然后，用户设置数据采集时间;

（5）然后，用户选择任意 2 个 Stop 通道，作为画出二维 Stop 时间差图像的通道。本例，选择通道 1 和通道 3;

（6）然后，开始采集。软件会实时采集和显示如下结果:

（6.1）通道 1/2/3 和通道 0 的时间差 Stop-start 统计直方图;

（6.2）通道 1/2/3 和通道 0 的时间差 Stop-start 统计直方图平均后的结果;

（6.3）通道 1 和通道 3 分别与通道 0 时间差 Stop-start 的关系图像（横坐标为 ch1，纵坐标为 ch3）。所有结果都可以保存。

（7）本例中，软件会统计每次 Start 信号后面所有的 Stop 信号。将时间差记作 Δ*t*(通道，*start* 编号，*stop* 编号）。上图中软件会将 Δ*t*(*c*h1,0,0)/Δ*t*(*c*h1,0,1)/Δ*t*(*c*h1,0,2)/Δ*t*(*c*h1,1,0)/Δ*t*(*c*h1,1,1)/Δ*t*(*c*h1,2,0) 统计到通道 1 的分布直方图。其他通道方法相同。

（8）用户也可以采集全局符合原始数据来自行处理。

另外，用户可以采集和保存 Start-stop 时间差数据。首先，用户应该先配置 Start 和 Stop 通道分组，然后框选“保存 T3 数据”，然后在软件安装目录下的 Output 文件夹下会保存 Start-stop 时间差数据，包括 Start 信号序列、Start 通道号、Stop 通道号、Start 与 Stop 时间差。



	A	B	C	D	E
1	start_channel_index	start_channel	stop_channel	timediff(ps)	
2	49993	4	5	939	
3	49993	4	7	953	
4	49993	4	6	995	
5	50012	0	2	912	
6	50012	0	3	997	
7	50012	0	1	1016	
8	49994	4	5	939	
9	49994	4	7	949	
10	49994	4	6	978	
11	50013	0	2	916	
12	50013	0	3	1003	
13	50013	0	1	1017	
14	49995	4	5	931	
15	49995	4	7	955	
16	49995	4	6	970	
17	50014	0	2	912	
18	50014	0	3	998	
19	50014	0	1	1020	
20	49996	4	5	933	
21	49996	4	7	935	
22	49996	4	6	979	
23	50015	0	2	912	
24	50015	0	3	994	
25	50015	0	1	1020	
26	49997	4	5	934	
27	49997	4	7	935	
28	49997	4	6	980	
29	50016	0	2	918	
30	50016	0	3	1002	
31	50016	0	1	1016	
32	49998	4	7	935	
33	49998	4	5	942	
34	49998	4	6	988	
35	50017	0	2	912	
36	50017	0	3	999	

图 59 Start-stop 输出 T3 数据的格式示意图（包含 Start 触发序列、start 通道号、stop 通道号、时间差信息）

（1）Start 信号序列：一般试验中，Start 为探测器同步（Sync）信号，通过产生多次 Start 信号来激发 Stop 信号（探测器输出）。Start 信号序列为 Start 信号发生的序列。这个序列值有利于用户去区分哪一次 Start 信号所激发的 Stop 信号；

（2）Start/stop 通道号可以让用户区分不同分组的 Start-stop 数据；

（3）时间差为在用户所设置时间窗内，Stop-Start 时间差值



图 60 双通道 Stop-Start 时间差热点图

此外，Venus 还支持用户在线实时观察双 stop 通道与 start 通道时间差的分布热点图，如上图所示。用户只需要选择哪一分组，分组中哪两个 stop 通道，即可实时观察时间差二维分布热点图。

一个常见的应用如下图所示。信号发生器输出两路同步信号，一路给光源（如激光器等）激发输出光信号，然后经过光路后打入测试样品上，测试样品激发输出的光信号进入光电探测器（如 PMT 等）后转换为电信号。将该电信号（可作为 Stop 信号）与同步参考信号（可作为 Start 信号）输入到 Venus 设备中，然后使用本 Start-stop 分析功能就可以通过连续测量得到不同时间的光子探测效率等数据，用于进行材料分析等。

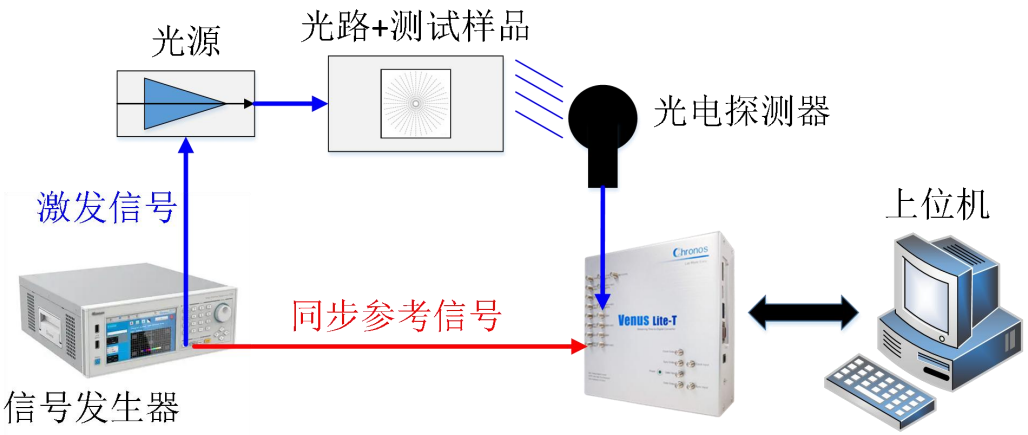


图 61 一种常见的 Start-stop 功能应用测试场景

8.7.4 ADC 波形

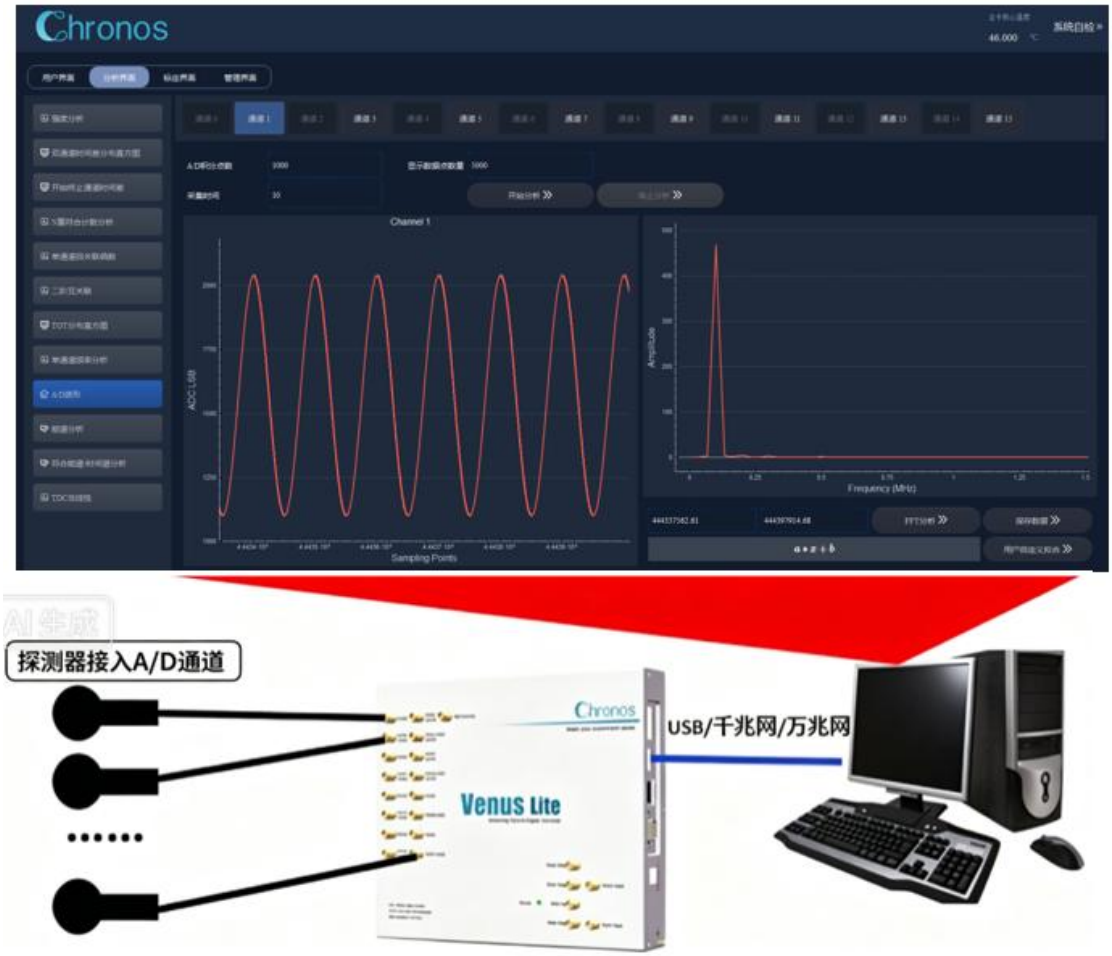


图 62 ADC 瞬态波形分析

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 63 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

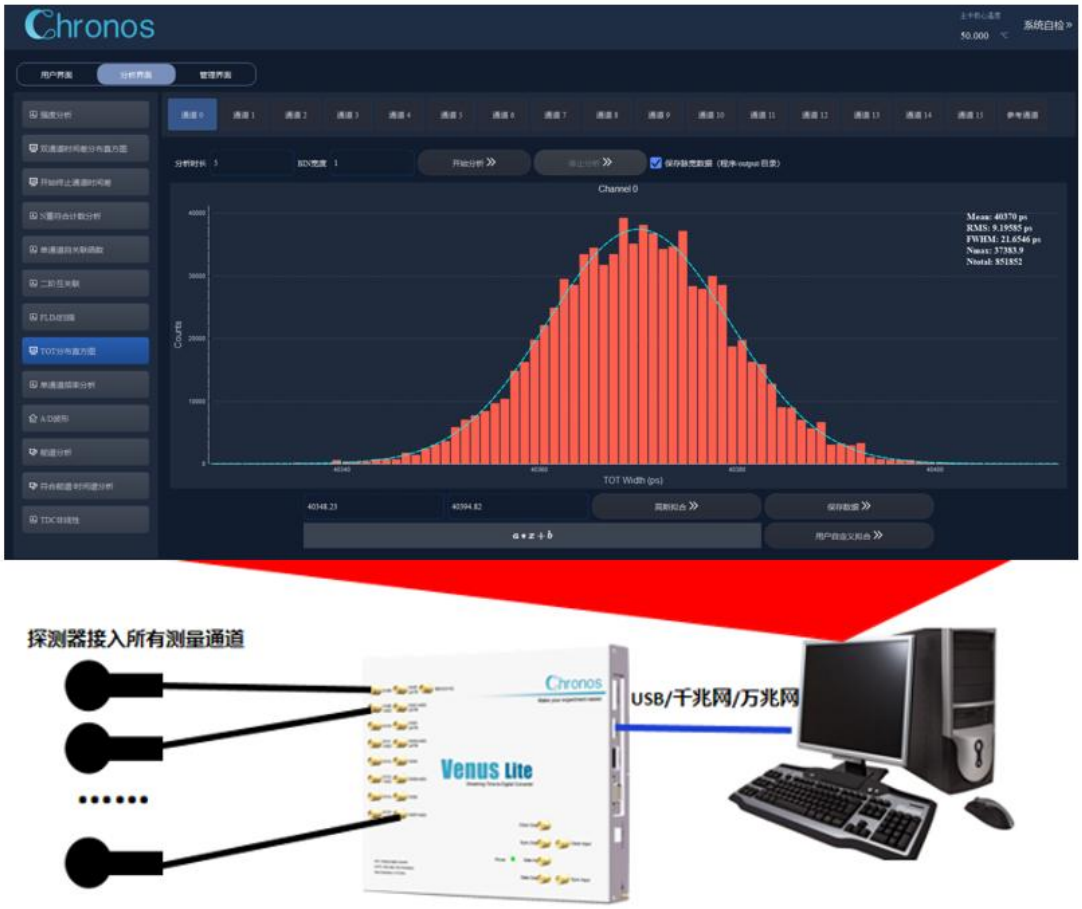


图 64 TOT 脉宽分布分析

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。



	A	B
1	channel	tot_ps
2	0	40371
3	0	40365
4	0	40359
5	0	40376
6	0	40366
7	0	40368
8	0	40361
9	0	40378
10	0	40373
11	0	40379
12	0	40354
13	0	40376
14	0	40345
15	0	40376

图 65 TOT 分析过程数据保存

此外，用户还可以在分析过程中保存脉宽时间数据，只需要勾选“保存脉宽数据”，数据就可以在安装目录下的 output 文件夹下。

8.7.7 符合能谱-时间谱分析

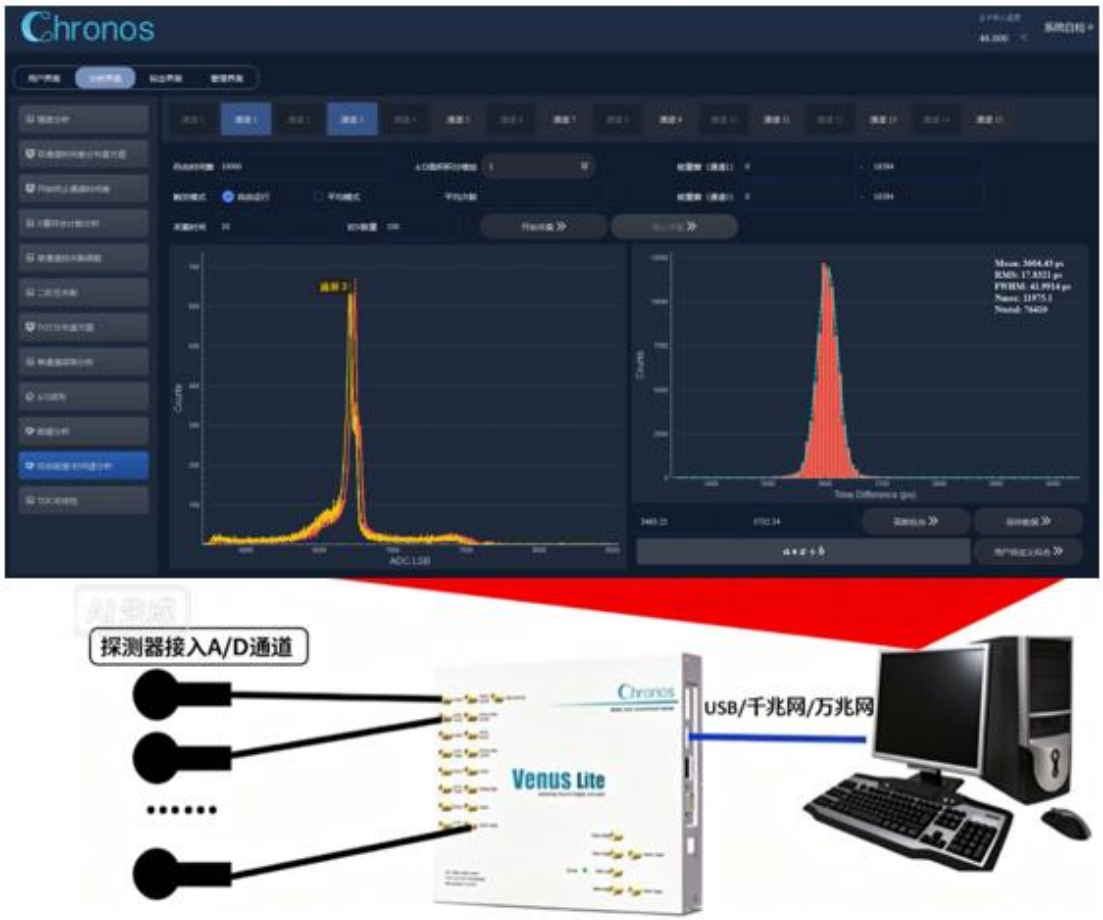


图 66 符合能谱-时间谱分析

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。

一种常见的符合测试场景如下图所示。利用闪烁体探测器(如 BGO/LYSO 等)

将伽马放射源（生成一对能量相同，方向相反，时间同步的伽马射线源）转换为电脉冲信号，然后通过光电探测器（如 PMT/SiPM 等）测量信号的能谱和到达时间。用于闪烁体材料测试、放射源检测、光电探测器性能研究等等。

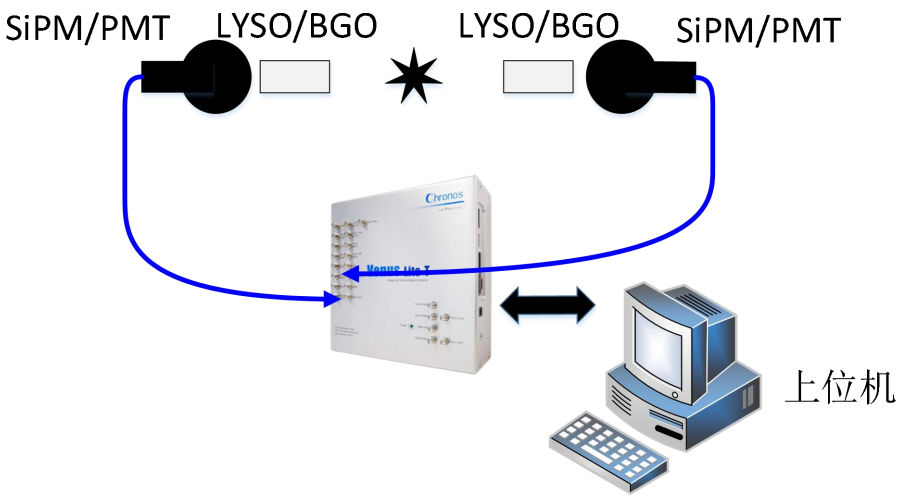
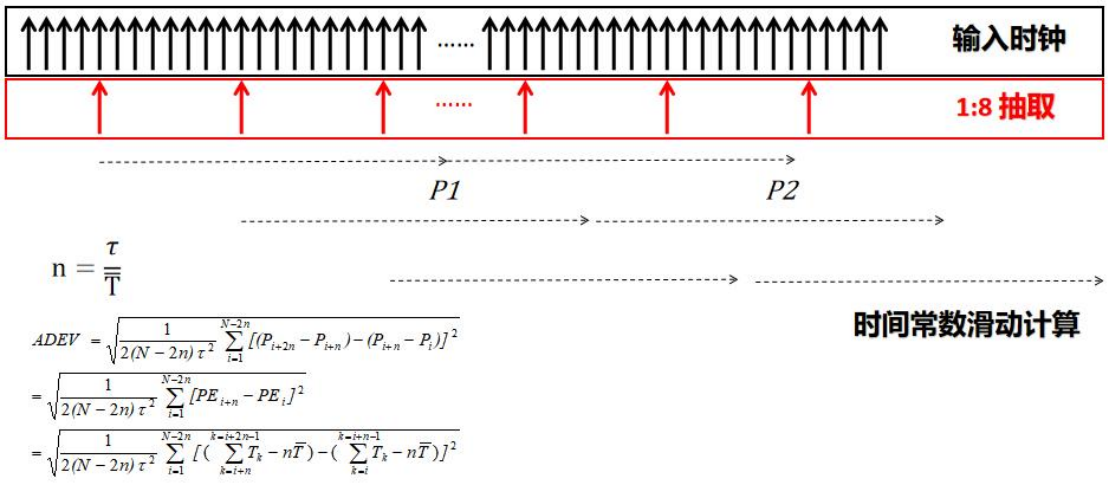


图 67 一种典型的能谱-时间符合测试平台

8.7.8 频率分析





$$n = \frac{\tau}{\overline{T}}$$

$$ADEV = \sqrt{\frac{1}{2(N-2n)\tau^2}\sum_{i=1}^{N-2n}[(P_{i+2n}-P_{i+n})-(P_{i+n}-P_i)]^2}$$

$$= \sqrt{\frac{1}{2(N-2n)\tau^2}\sum_{i=1}^{N-2n}[PE_{i+n}-PE_i]^2}$$

$$= \sqrt{\frac{1}{2(N-2n)\tau^2}\sum_{i=1}^{N-2n}[(\sum_{k=i+n}^{k=i+2n-1}T_k - n\overline{T}) - (\sum_{k=i}^{k=i+n-1}T_k - n\overline{T})]^2}$$

图 68 单通道频率分析

对于频率分析，目前只支持对通道 0-1 有效，且每次测试只能接其中的 1 个通道。注意：为了防止其他通道数据干扰，建议用户关闭其他通道（参见 8.6.1 节）。

如上图所示，Venus 首先会对输入信号进行 1:8 提取，然后计算时钟的修正艾伦偏差（MDEV）、哈达玛偏差（HDEV）、时间偏差（TDEV）和艾伦偏差（ADEV）等信息，并计算这些偏差与时间常数的关系，这些方差的计算公式见附录 G。

以 ADEV 计算为例对计算过程进行介绍：

1. 首先软件计算输入时钟的平均周期或者使用用户指定的参考周期$\bar{T}$;
2. 然后，软件会先计算 n=1 时的每个时间常数边缘的相位误差（$\sum T_i - n\bar{T}$）；
3. 然后，软件会逐一滑动计算 n=1 时的相位误差值，一共计算 N-2n 次，然后带入 ADEV 公式进行计算，得到 ADEV 值;
4. 自动重复计算不同τ下的 ADEV 值，得到 ADEV 与τ的关系并实时显示。

在使用时，用户需设置数量级范围、每数量级点数、平均次数、分析数量、FE 分析周期数、标称频率、BIN 数量、采集时间等，随后点击开始采集。软件会实时计算输入这些参数设置说明如下：

- 采集时间：数据采集时长，单位 s;
- 分析数量：频率稳定性分析每次计算偏差的总周期数据数目;
- 平均次数：频率稳定性分析每次计算偏差的次数（如上图中的 N-2n）；
- 标称频率：频率稳定性分析用户可以指定输入时钟的频率，如果不填写，软件会自动计算输入时钟的平均频率($\bar{T}$);
- 数量级范围：频率稳定性分析τ显示范围分为几个十倍程(decades);
- 每数量级点数：频率稳定性分析每个数量级范围内显示的点数;
- FE 分析点数：即每次分析和显示多少个时钟周期并进行瞬时频率误差和累计误差进行分析;
- BIN 数量：周期分布直方图中的时间轴 BIN 的数目。

对于时钟周期，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、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。

8.7.9 二阶自关联函数（ $g^{(2)}$）分析

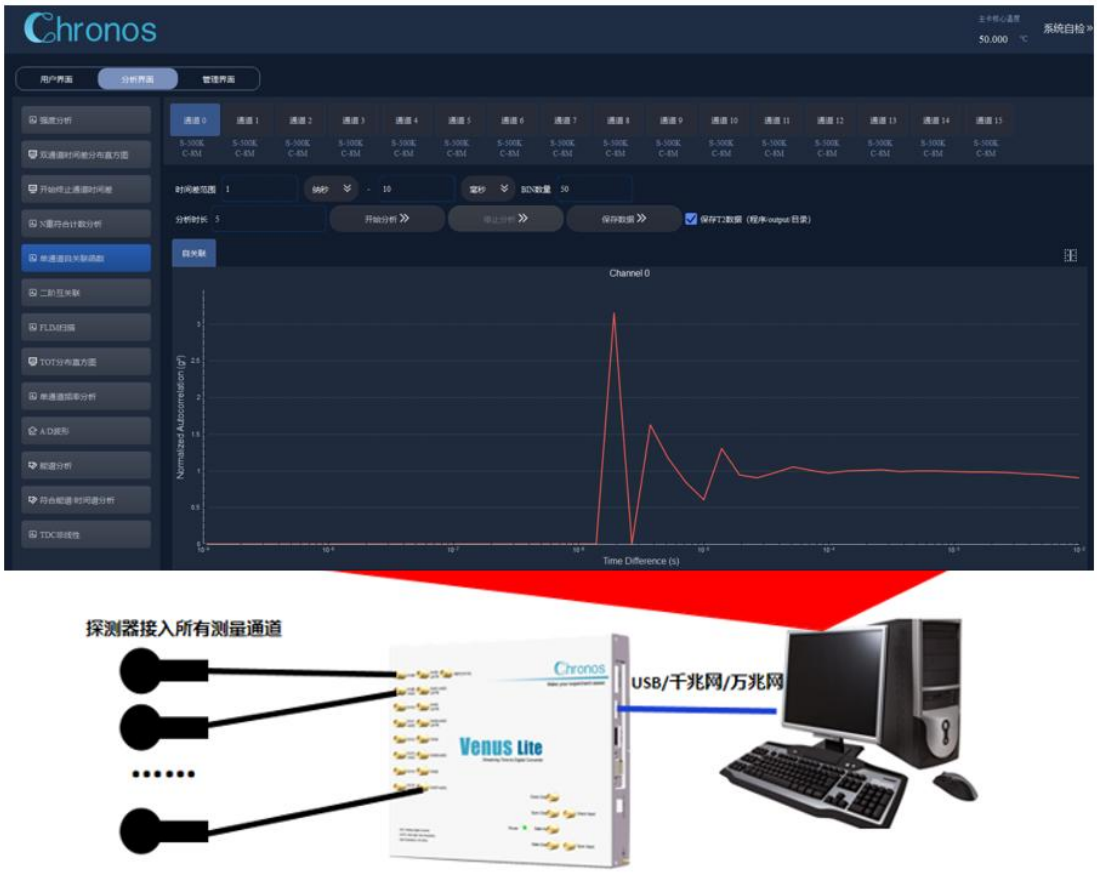


图 69 二阶自关联函数分析

本分析功能模块旨在对单光子计数实验产生的流式时间戳数据进行实时处理，计算单通道的二阶强度自关联函数 $g^{(2)}(\tau)$。关于其数学原理，见附录 H。用户需要对以下参数进行设置。

- 时间差范围：定义分析的最小时间差和最大时间差。最大时间差上限是 1 秒内；
- 边沿分析：在统计计数时间戳的时候，需要判断边沿类型，用户指定是上升沿还是下降沿；
- 采集时间：定义数据采集的时间，单位 s，到时间后自动停止。



	A	B	C
1	channel	timestamp_ps	
2	0	737808782464498	
3	0	737808784464511	
4	0	737808786464514	
5	0	737808788464505	
6	0	737808790464512	
7	0	737808792464518	
8	0	737808794464515	
9	0	737808796464517	
10	0	737808798464527	
11	0	737808800464514	
12	0	737808802464518	
13	0	737808804464520	
14	0	737808806464516	
15	0	737808808464522	

图 70 保存 T2 时间戳数据

此外，Venus 还支持采集过程中 T2 时间戳数据的保存，只需要勾选“保存 T2 数据”，数据就可以在安装目录下的 output 文件夹下。

8.7.10 TDC 非线性测试



图 71 TDC 非线性测试

Venus TDC 采用全自动非线性校准技术，在如下条件下，系统发起自动校准功能：

（1） 系统上电；

（2） 系统复位；

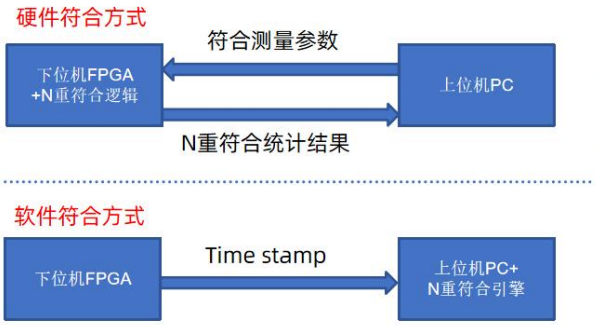
（3） 系统 TDC 参考时钟来源切换。

本分析功能模块旨在对 Venus 设备的核心 TDC 逻辑的静态指标微分非线性 DNL 和积分非线性 INL 进行测试。用户只需要输入采集时间和通道选择，即可分析某通道 TDC 的非线性指标。此功能作为用户评估当前 TDC 性能的工具。

8.7.11 多重符合统计

Venus N 重符合包括硬件符合或者软件符合，用户可以通过上位机软件和实验需求来选择使用哪种方式。

Venus TDC N 重符合分为两类，一类是将 N 重符合引擎部署在设备 FPGA 芯片中，FPGA 芯片会接收来自上位机软件的符合策略参数，然后实时地、低延迟地、高效地并行对 40 个策略组进行 N 重符合统计，并将统计结果传输给上位机软件。上位机软件将统计计数率结果实时显示；另外一类是将 N 重符合引擎部署在上位机电脑上，上位机电脑接收来自 TDC 的时间戳 Raw 数据，然后统计 N 重符合计数率和符合中间结果。其架构如下所示。



性能参数	硬件N重符合	软件N重符合
最大策略组数	40	200
每组通道配置数	16	16
每组虚拟时间窗	0-63 ns	0-63 ns
总体符合计数率	<100 Mtags/s	依赖上位机电脑性能
输出内容	各策略组计数率	各策略组计数率+符合过程数据
统计延迟	低，策略部署完成后下1s即更新	大，依赖上位机电脑性能

图 72 两种 N 重符合机制

可见，相比软件实现，硬件实现 N 重符合具有低延迟、高效率、高性能的优势，如果用户只需要统计计数率结果，而不需要符合中间数据，那么建议采用硬件符合方式；软件 N 重符合高度依赖上位机电脑性能，且其延迟大，但其可以部署更多策略数和输出符合过程数据。所以给出如下建议：

（1）如果用户只需要 N 重符合计数率结果，建议使用硬件符合机制；

（2）如果用户上位机电脑性能较好，输入计数率较低（比如<10 Mtags/s total），需要部署超过 40 个策略组，或需要中间符合过程数据，那么可以选择软件符合机制。

上位机软件进行 N 重符合页面如下所示。

（1）用户可以选择“软件”还是“硬件”分析模式；

（2）用户可以新建/修改/删除符合策略组；

（3）用户设置完策略组后，需要点击“配置策略”，如果需要保存统计数据，选择“开始保存”；

（4）支持设置计数率窗口（范围：10ms - 10s），和强度分析页面联动。

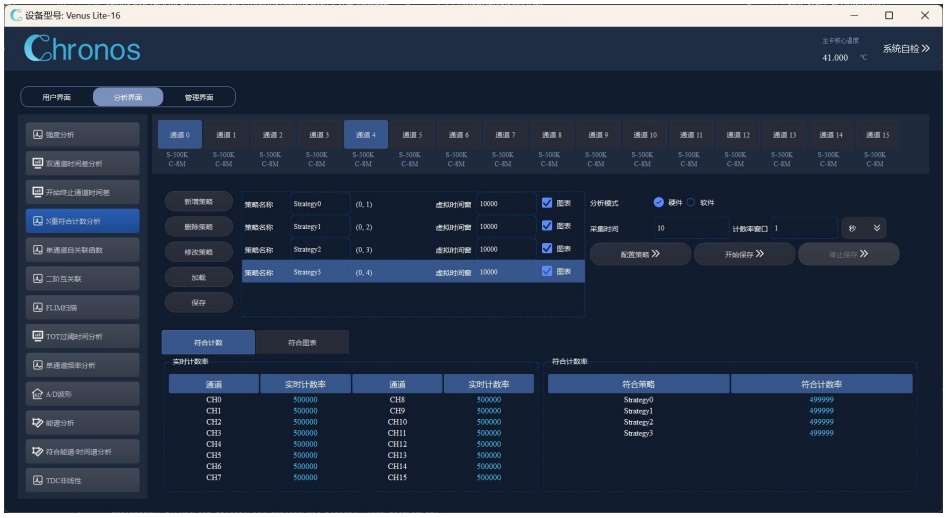


图 73 N 重符合上位机软件示意图

此外，为了方便用户及时保存和载入多个符合策略，软件支持用户保存当前测量配置文件，以及加载策略配置文件，其格式如下所示。

```
NAME        CHANNELS  TWIN
Strategy0   0_1     10000
Strategy1   1_2     10000
Strategy2   2_3     10000
Strategy3   3_4     10000
Strategy4   3_4_5  10000
Strategy5   4_5_6  10000
Strategy6   5_6_7  10000
Strategy7   6_7_8  10000
Strategy8   7_8_9  10000
Strategy9   8_9_10       10000
Strategy10  9_10_11      10000
Strategy11  10_11_12     10000
Strategy12  11_12_13     10000
Strategy13  12_13_14     10000
Strategy14  13_14_15     10000
Strategy15  0_13_14_15  10000
Strategy16  0_1_13_14_15       10000
Strategy17  0_1_2_13_14_15     10000
Strategy18  0_1_2_3_13_14_15  10000
Strategy19  0_1_2_3_4_13_14_15       10000
Strategy20  0_1_2_3_4_5_13_14_15     10000
Strategy21  0_1_2_3_4_5_6_13_14_15  10000
Strategy22  0_1_2_3_4_5_6_7_13_14_15       10000
Strategy23  0_1_2_3_4_5_6_7_8_13_14_15     10000
Strategy24  4_5_6_7_8_13_14_15       10000
Strategy25  5_6_7_8_13_14_15  10000
Strategy26  6_7_8_13_14_15     10000
Strategy27  7_8_13_14_15       10000
Strategy28  8_13_14_15  10000
Strategy29  8_9_13_14_15       10000
Strategy30  9_10_13_14_15      10000
Strategy31  10_11_13_14_15     10000
Strategy32  11_12_13_14_15     10000
Strategy33  0_11_12_13_14_15  10000
Strategy34  0_1_11_12_13_14_15       10000
Strategy35  0_1_2_11_12_13_14_15     10000
Strategy36  0_1_2_3_11_12_13_14_15  10000
Strategy37  0_1_2_3_4_11_12_13_14_15       10000
Strategy38  0_1_2_3_4_5_11_12_13_14_15     10000
Strategy39  0_1_2_3_4_5_6_11_12_13_14_15  10000
```

图 74 策略配置文件

保存的统计结果包括各通道的计数率以及各个符合策略组的符合计数率，如

下所示。

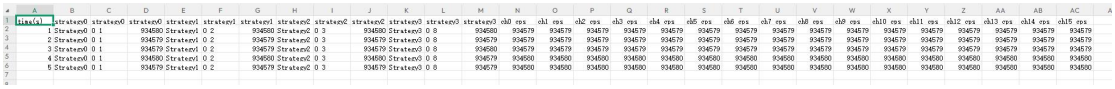


图 75 统计结果保存格式

8.7.11.1 硬件在线 N 重符合统计

多重符合原理示意图如下所示。只有当所有指定通道的触发时间都在各自时间窗之内才作为有效符合计数。

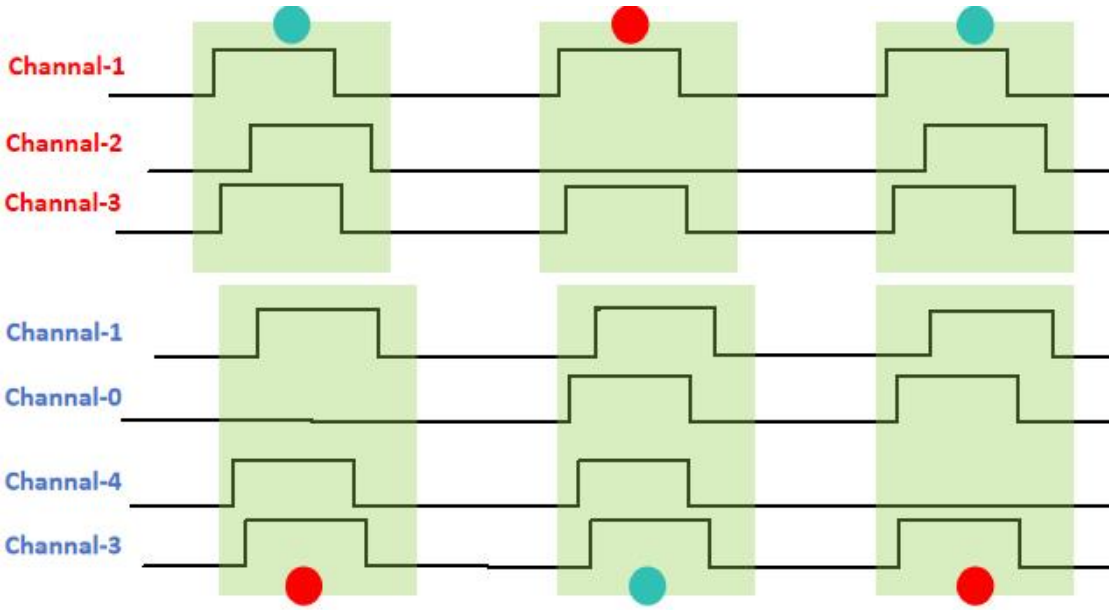


图 76 多重符合示意图（以 2 个策略组，3/4 重符合为例）

首先用户需要部署多组 N 重符合策略，即指定最大不超过 40 组的符合通道组（简称虚拟通道），以及各组的虚拟时间窗。点击开始分析，上位机软件会将符合策略参数下发给设备，Venus 会实时统计各个策略组的多重符合计数值并显示和保存。

8.7.11.2 软件在线 N 重符合统计

软件在线 N 重符合与硬件方式的配置方式基本相同，首先用户需要部署多组 N 重符合策略，即指定最大不超过 200 组的符合通道组（简称虚拟通道），以及

各组的虚拟时间窗。点击开始分析，Venus 会对符合数据进行实时分析以及输出多重符合数据的符合事件序列 index、双符合通道号和双符合通道的时间差信息，作为中间输出数据文件给用户。用户直接可以把此文件内容作为 N 重符合事件的时间信息进行下一步的分析。

比如，用户建立{CH0, CH1, 15ps}, {CH1,CH2,CH3,CH4,200ps},{CH4,CH5,CH6,CH7,10000ps}三个虚拟通道分组，那么软件会实时计算 CH0、CH1 时间差小于 15ps 的组 1 符合计数率，实时计算 CH1、CH2、CH3 和 CH4 之间时间差小于 200ps 的组 2 符合计数率，实时计算 CH4、CH5、CH6 和 CH7 之间时间差小于 10000ps 的组 3 符合计数率并显示计数率与采集时间关系的曲线。



	A	B	C	D	E	F
1	strategy_name	coins_index	start_ch	start_timestamp(ps)	end_ch	end_timestamp(ps)
2	Strategy0	1	0	182320024358104	1	182320024358086
3	Strategy0	1	0	182320024358104	2	182320024357982
4	Strategy0	2	0	182320026358104	1	182320026358086
5	Strategy0	2	0	182320026358104	2	182320026357982
6	Strategy0	3	0	182320028358098	1	182320028358086
7	Strategy0	3	0	182320028358098	2	182320028357966
8	Strategy0	4	0	182320030358099	1	182320030358086
9	Strategy0	4	0	182320030358099	2	182320030357979
10	Strategy0	5	0	182320032358098	1	182320032358086
11	Strategy0	5	0	182320032358098	2	182320032357966
12	Strategy0	6	0	182320034358099	1	182320034358086
13	Strategy0	6	0	182320034358099	2	182320034357973

图 77 符合分析结果是否保存以及保存文件格式

另外，用户还可以选择保存实时分析符合结果，结果保存在 Venus 程序安装目录下的/output 文件夹下。保存文件格式为 csv，第一列为分组名称，第二列为符合事件的编号，即哪一次符合事件；Start_ch、End_ch、Start_timestamp(ps)、End_timestamp(ps)为两个符合通道的通道号以及各自的时间戳信息（单位 ps）。

举例说明，虚拟分组 CH0/1/2，发生了第一次符合事件，见上图中 csv 文件中的 2-3 行，用户可以简单观测到发生本次符合事件的 3 个通道的时间戳信息。

8.7.12 分组二阶互关联系数（g_{AB}^2)



图 78 分组二阶互关联系数计算

Venus GUI 支持多组任意两个通道的互关联系数g_{AB}^2的同时实时计算，其公式参考附录 I。在计算g_{AB}^2时，用户需要输入两个通道的暗计数值。用户可以先评估探测器的暗计数值，如果用户忽略暗计数值，不输入即可，软件计算时则默认暗计数值为 0。



	A	B	C	D	E
1	index	channel_a	channel_b	timediff_ps(Tb-Ta)	
2	50014	0	1	985	
3	50014	0	2	1022	
4	50015	0	1	991	
5	50015	0	2	1030	
6	50016	0	1	987	
7	50016	0	2	1030	
8	50017	0	1	985	
9	50017	0	2	1026	
10	50018	0	1	989	
11	50018	0	2	1030	
12	50019	0	1	987	
13	50019	0	2	1030	
14	50020	0	1	983	
15	50020	0	2	1026	
16	50021	0	1	991	
17	50021	0	2	1034	
18	50022	0	1	980	

图 79 二阶互关联分析中过程 T3 数据保存

另外，Venus 还支持采集过程中 T3 时间差数据的保存，只需要勾选“保存 T3 数据”，数据就可以在安装目录下的 output 文件夹下。

8.7.13 FLIM 扫描测试

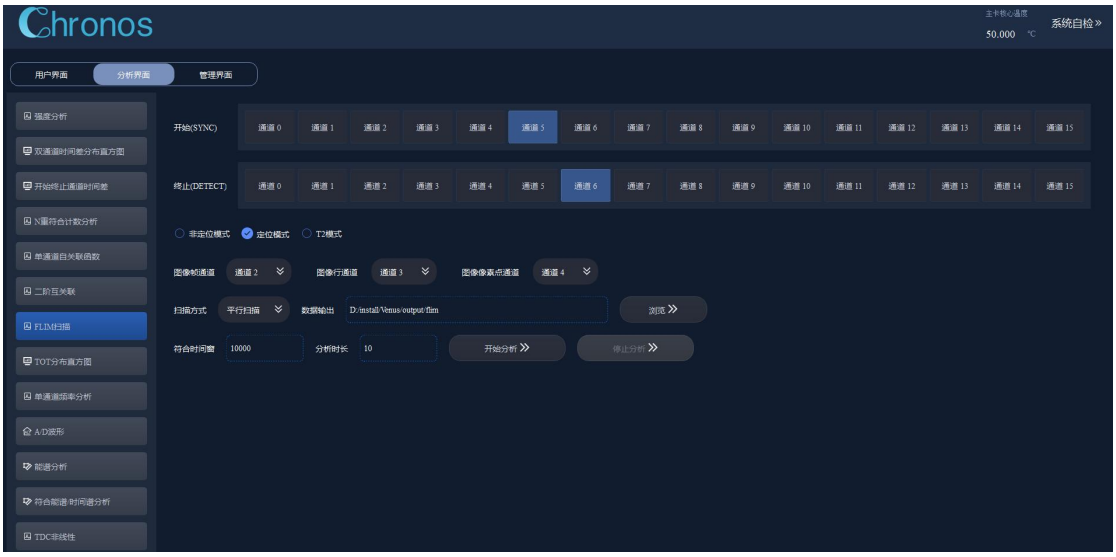


图 80 FLIM 扫描软件功能

Venus 支持在线高速荧光寿命成像显微镜（Fluorescence Lifetime Imaging Microscopy）扫描测试。包括三种方式供用户选择：

（1）定位模式：用户输入位置(Frame/Line/Pixel)通道、Sync 通道和 Signal 通道信号。软件自动输出位置信息，以及 Sync 和 Signal 时间差信息；

（2）非定位模式：用户仅输入 Sync 通道和 Signal 通道信号。软件输出 Sync 和 Signal 时间差信息，以及每次物理符合事件发生时的时间戳信息；

（3）T2 模式：用户仅输入 Sync 通道和 Signal 通道信号。软件输出 Sync 和 Signal 时间戳信息。完全由用户自行处理数据；

8.7.13.1 定位模式

首先，用户应该先配置位置（Frame/Line/Pixel）通道、Sync 通道和 Signal 通道：

（1）Frame 通道：FLIM 图像帧切换指示信号；

（2）Line 通道：FLIM 每一 Frame 图像中，每行扫描切换指示信号；

（3）Pixel 通道：FLIM 每一 Line 图像中，每个 Pixel 扫描切换指示信号；

（4）Sync 通道：为探测器激发使能同步信号；

（5）Signal 通道：为荧光激发探测器输出的电信号。

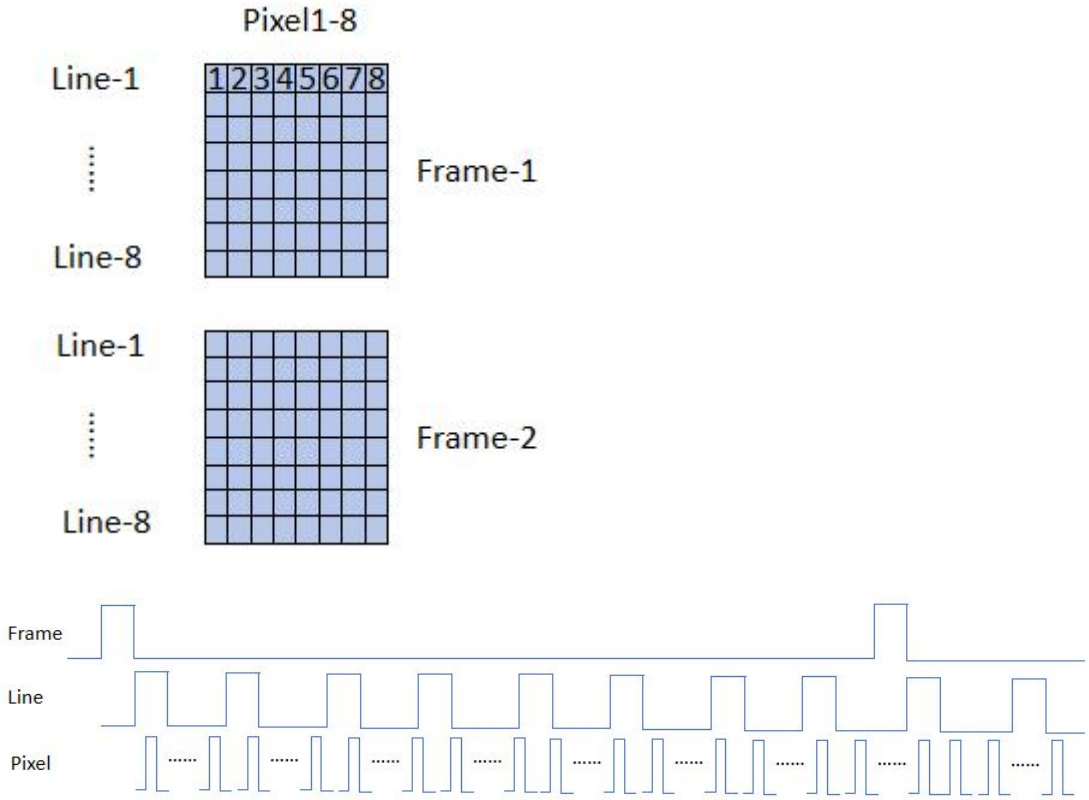


图 81 同步扫描位置信号（Frame/Line/Pixel 示意图，以 2 帧，8x8 图像为例）

Venus 扫描支持高达 262144x262144 大小的图像，足以满足应用需求。

然后，用户按照配置通道内容，将 Frame、Line、Pixel 位置指示信号，以及 Sync 和 Signal 信号接入相应通道。然后点击开始采集，Venus 将输出 FLIM 数据，如下，包括：

（1）Frame 的时间戳信息；

（2）Line 的时间戳信息；

（3）Pixel 位置计数值、Sync 通道号、Signal 通道号以及 Signal 与 Sync 信号时间差信息。

用户通过以上位置和时间差信息，结合扫描序列数据，可以进行进一步拟合得到 FLIM 图像。用户可以选择平行扫描或者蛇形扫描方式进行采集。因此，FLIM 扫描需要 5 个输入信号。得到的扫描数据存放在软件安装目录下的 output 文件夹内，其数据格式如下所示。用户通过时间差信息拟合得到寿命信息。

	A	B	C	D
1	frame_timestamp_ps	line_timestamp_ps	pixel_index	timediff_ps
2	0	0	301	3969
3	0	0	301	3973
4	0	0	301	3968
5	0	0	301	3970
6	0	0	301	3976
7	0	0	301	3978
8	0	0	301	3971
9	0	0	301	3970
10	0	0	301	3984
11	0	0	301	3981
12	0	0	301	3981
13	0	0	301	3970
14	0	0	301	3976
15	0	0	301	3982
16	0	0	301	3973
17	0	0	301	3974
18	0	0	301	3975
19	0	0	301	3981

图 82 FLIM 定位扫描模式下的数据输出格式

8.7.13.2 非定位模式



图 83 FLIM 非定位模式

首先，用户应该先配置 Sync 通道和 Signal 通道：

（1）Sync 通道：为探测器激发使能同步信号；

（2）Signal 通道：为荧光激发探测器输出的电信号。

然后，用户设置一个时间窗和采集时长，就在保存目录下得到 T3 时间差数据以及每次事件发生的时间戳信息。用户根据时间戳信息来计算扫描位置，通过时间差信息拟合得到寿命信息。

	A	B	C	D	E
1	timestamp_ps	start_ch	stop_channel	timediff_ps	
2	100009780000	2	3	23	
3	100009790000	0	1	15	
4	100011780000	2	3	31	
5	100011790000	0	1	15	
6	100013780000	2	3	31	
7	100013790000	0	1	15	
8	100015780000	2	3	39	
9	100015790000	0	1	7	
10	100017780000	2	3	39	
11	100017790000	0	1	15	
12	100019780000	2	3	31	
13	100019790000	0	1	15	
14	100021780000	2	3	23	
15	100021790000	0	1	15	
16	100023780000	2	3	31	
17	100023790000	0	1	15	
18	100025780000	2	3	31	
19	100025790000	0	1	15	

图 84 FLIM 非定位扫描模式下的数据输出格式

8.7.13.3 T2 模式

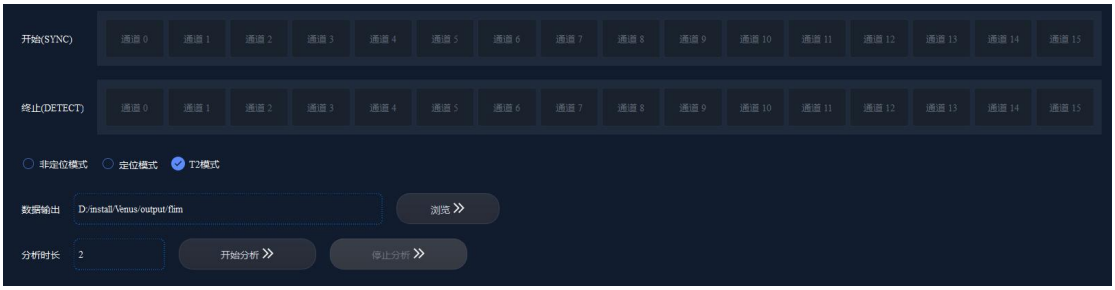


图 85 FLIM 中 T2 模式

T2 模式即保存原始流时间戳信息，是最简单和最完整的时间信息。用户拿到每个时间信息然后计算扫描位置和寿命信息。

	A	B	C
1	channel	timestamp_ps	
2	10	1896484878978626	
3	11	1896484878978599	
4	12	1896484878978735	
5	13	1896484878978686	
6	14	1896484878978698	
7	15	1896484878978697	
8	16	1896484878978695	
9	0	1896484878978828	
10	1	1896484878979822	
11	2	1896484878979853	
12	3	1896484878979911	
13	4	1896484880979689	
14	5	1896484880979699	
15	6	1896484880978722	
16	7	1896484880978572	
17	8	1896484880978576	
18	9	1896484880978688	
19	10	1896484880978626	
20	11	1896484880978600	
21	12	1896484880978735	
22	13	1896484880978686	
23	14	1896484880978707	

图 86 FLIM 中 T2 模式下的数据输出格式

8.9 管理页面

8.9.1 基本介绍

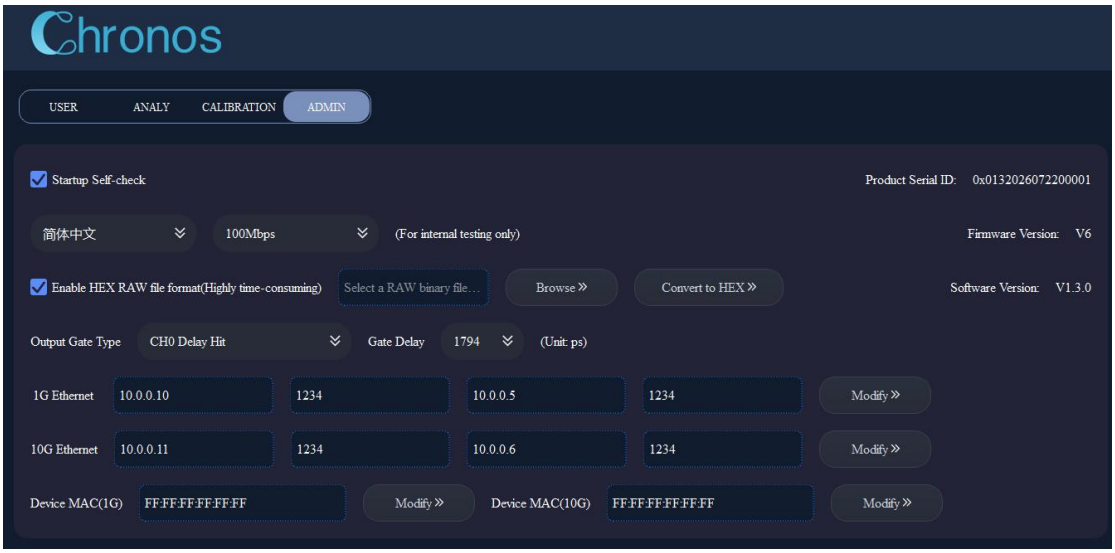


图 87 管理页面

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
- 辅助 Gate 输出

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	
段 3	生产日期	例如：0x20251101
段 4	生产地点	例如：0x0，杭州
段 5	生产编号	例如：0x1

8.9.2 辅助 Gate 输出

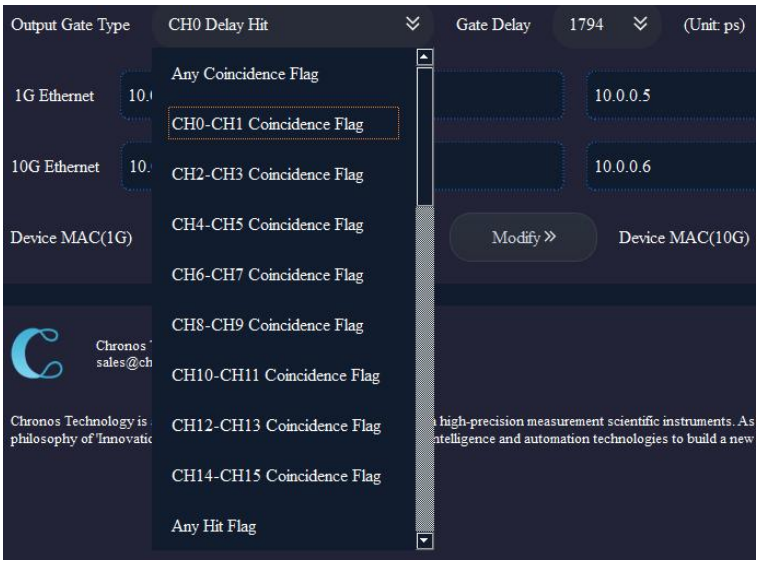


图 88 Gate 输出功能选择

辅助 Gate 输出，用户通过配置不同功能定义 Output Gate SMA 输出，如下表所述。

表格 8 Gate 输出功能描述

序号	软件配置	意义	说明
1.	全局符合标志	任意 2 个通道符合事件发生，输出高电平脉宽	一次符合事件输出脉冲宽度约 10 ns，如果连续多次符合，输出脉冲宽度为 10 x N ns。
2.	通道 0 和通道 1 符合标志	通道 0 和通道 1 符合事件发生，输出高电平脉宽	
3.	通道 2 和通道 3 符合标志	通道 2 和通道 3 符合事件发生，输出高电平脉宽	
4.	通道 4 和通道 5 符合标志	通道 4 和通道 5 符合事件发生，输出高电平脉宽	
5.	通道 6 和通道 7 符合标志	通道 6 和通道 7 符合事件发生，输出高电平脉宽	
6.	通道 8 和通道 9 符合标志	通道 8 和通道 9 符合事件发生，输出高电平脉宽	
7.	通道 10 和通道 11 符合标志	通道 10 和通道 11 符合事件发生，输出高电平脉宽	
8.	通道 12 和通道 13 符合标志	通道 12 和通道 13 符合事件发生，输出高电平脉宽	

9.	通道 14 和通道 15 符合标志	通道 14 和通道 15 符合事件发生，输出高电平脉宽	
10.	通道触发标志	任意一个通道有触发发生后，输出高电平脉宽	一次事件输出脉冲宽度约 10 ns，如果连续多次触发，输出脉冲宽度为 10 x N ns。
11.	通道 0 触发标志	通道 0 有触发发生后，输出高电平脉宽	
12.	通道 1 触发标志	通道 1 有触发发生后，输出高电平脉宽	
13.	通道 2 触发标志	通道 2 有触发发生后，输出高电平脉宽	
14.	通道 3 触发标志	通道 3 有触发发生后，输出高电平脉宽	
15.	系统时钟锁定		低电平表示系统正常
16.	Input gate		转发 input gate 输入信号，用于级联硬件数据采集（8.6.8 节）
17.	通道 0-通道 16 触发脉冲延迟输出	通道 0-通道 16 触发脉冲经过可控延迟后输出	延迟值 0~2.5 ns，最小分辨 78 ps
18.	N 重符合第 0 组策略有效输出	N 重符合模式，第 0 组策略有效时输出脉冲	一次事件输出脉冲宽度约 10 ns，如果连续多次触发，输出脉冲宽度为 10 x N ns。

用户通过管理页面中的 Output Gate Type 来选择功能。当选择通道 0-通道 16 触发脉冲延迟输出时，通过 Gate Delay 选择输出脉冲相对于输入脉冲的延迟，延迟范围为 2.5 ns，最小分辨为 78 ps。

8.10 日志区

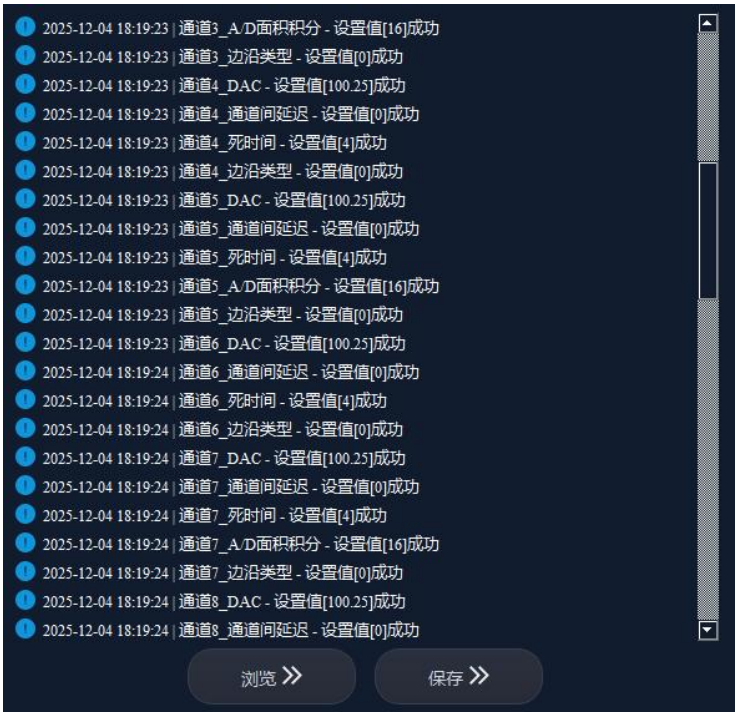


图 89 系统日志

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历史信息。

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 获取设备接口列表后，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。

【设备接口信息结构体-cinst_device_info_t】

字段	类型	说明
type	int32	接口类型
name	char[256]	接口名称
serial_number	char[16]	USB3 设备序列号

【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-设备接口数量	获取可用设备接口列表	设备信息包含类型、名称和序列号等； 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-设备接口数量	获取可用设备接口数量	
ci_get_device（ci_handle_t handle, int32_t index, int32_t* out_type, char* out_name, int32_t name_size, char* out_sn, int32_t sn_size）	handle：对象句柄 Index：设备 index out_type:设备接口类型 out_name：设备接口名称 name_size：名称缓存大小 out_sn：设备接口序列号 sn_size：序列号缓存大小	int - 状态码	获取指定设备信息（包括：接口类型、名称和序列号）	仅 USB3 接口类型支持设备序列号
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	连接设备（指定接口类型）	具体类型定义请详见 ConnType 定义。

ci_connect(ci_handle_t handle, const char* device_sn)	handle：对象句柄 device_sn：设备接口序列号	int - 状态码	连接设备（指定设备接口序列号）	仅支持 USB3 设备
disconnect(ci_handle_t handle)	handle：对象句柄	int - 状态码	断开当前设备连接	

9.1.4 错误处理

【错误码定义-cinst_error_t】

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

【API 定义】

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

错误回调函数定义：

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

9.1.5 设备自检

设备使用前，可以先进行设备自检操作，设置各通道和全局配置的初始默认值。

设备自检后，在 selfcheck 目录下会生成自检日志文档，可以查看自检状态。

【API 定义】

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

9.1.6 通道配置

注意事项：

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

【阻抗类型定义-cinst_term_type_t】

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

【迟滞电压类型定义-cinst_hts_type_t】

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

【边沿触发类型定义-cinst_edge_type_t】

定义	值	说明
CINST_RISING_EDGE	0	上升沿

CINST_FALLING_EDGE	1	下降沿
CINST_PMIDDLE	2	正中间
CINST_NMIDDLE	3	负中间

【API 定义】

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

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

	针			

9.1.7 全局配置

注意事项：

- 测试模式类型主要用于仿真和测试，正常使用采用 RAW 模式；
- 切换 TDC 时钟来源会涉及到设备的重新校准，耗时较长（一般在 30s）；
- 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_25MHZ	1	外部时钟(25MHz)
CINST_EXTERNAL_CLOCK_10MHZ	2	外部时钟(10MHz)

【数据模式定义-cinst_data_mode_t】

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

CINST_DATA_MODE_GLOBAL_COINS_TIMESTAMP	12	全局符合（带时间戳）模式
CINST_DATA_MODE_SINGLE_TIMESTAMP	13	单边沿时间戳模式
CINST_DATA_MODE_GLOBAL_COINS2	14	全局符合模式 2

【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_INTERNAL_TRIGGER	6	内部触发模式

【数据带宽类型定义-cinst_test_data_bandwidth_t】

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

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

定义	值	说明
CINST_AREA_SCALE_1	0	1 倍
CINST_AREA_SCALE_1_2	1	1/2 倍
CINST_AREA_SCALE_1_4	2	1/4 倍
CINST_AREA_SCALE_1_8	3	1/8 倍
CINST_AREA_SCALE_1_16	4	1/16 倍
CINST_AREA_SCALE_1_32	5	1/32 倍

CINST_AREA_SCALE_1_64	6	1/64 倍
CINST_AREA_SCALE_1_128	7	1/128 倍

【数据采集方式定义-cinst_acq_type_t】

定义	值	说明
CINST_ACQ_SOFTWARE	0	软件方式
CINST_ACQ_HARDWARE	1	硬件方式

【输出门类型定义-cinst_output_gate_type_t】

定义	值	说明
CINST_GATE_TYPE_ANY_COINS	0	Any coincidence flag
CINST_GATE_TYPE_CH0_CH1_COINS	1	CH0 and CH1 coincidence flag
CINST_GATE_TYPE_CH2_CH3_COINS	2	CH2 and CH3 coincidence flag
CINST_GATE_TYPE_CH4_CH5_COINS	3	CH4 and CH5 coincidence flag
CINST_GATE_TYPE_CH6_CH7_COINS	4	CH6 and CH7 coincidence flag
CINST_GATE_TYPE_CH8_CH9_COINS	5	CH8 and CH9 coincidence flag
CINST_GATE_TYPE_CH10_CH11_COINS	6	CH10 and CH11 coincidence flag
CINST_GATE_TYPE_CH12_CH13_COINS	7	CH12 and CH13 coincidence flag
CINST_GATE_TYPE_CH14_CH15_COINS	8	CH14 and CH15 coincidence flag
CINST_GATE_TYPE_ANY_HIT	9	Any hit flag
CINST_GATE_TYPE_CH0_HIT	10	CH0 hit flag
CINST_GATE_TYPE_CH1_HIT	11	CH1 hit flag
CINST_GATE_TYPE_CH2_HIT	12	CH2 hit flag
CINST_GATE_TYPE_CH3_HIT	13	CH3 hit flag
CINST_GATE_TYPE_SYSTEM_CLOCK_PLL_LOCKED	14	System clock PLL locked
CINST_GATE_TYPE_I_GATE	15	i_gate
CINST_GATE_TYPE_CH0_DELAY_HIT	16	CH0 delay hit

CINST_GATE_TYPE_CH1_DELAY_HIT	17	CH1 delay hit
CINST_GATE_TYPE_CH2_DELAY_HIT	18	CH2 delay hit
CINST_GATE_TYPE_CH3_DELAY_HIT	19	CH3 delay hit
CINST_GATE_TYPE_CH4_DELAY_HIT	20	CH4 delay hit
CINST_GATE_TYPE_CH5_DELAY_HIT	21	CH5 delay hit
CINST_GATE_TYPE_CH6_DELAY_HIT	22	CH6 delay hit
CINST_GATE_TYPE_CH7_DELAY_HIT	23	CH7 delay hit
CINST_GATE_TYPE_CH8_DELAY_HIT	24	CH8 delay hit
CINST_GATE_TYPE_CH9_DELAY_HIT	25	CH9 delay hit
CINST_GATE_TYPE_CH10_DELAY_HIT	26	CH10 delay hit
CINST_GATE_TYPE_CH11_DELAY_HIT	27	CH11 delay hit
CINST_GATE_TYPE_CH12_DELAY_HIT	28	CH12 delay hit
CINST_GATE_TYPE_CH13_DELAY_HIT	29	CH13 delay hit
CINST_GATE_TYPE_CH14_DELAY_HIT	30	CH14 delay hit
CINST_GATE_TYPE_CH15_DELAY_HIT	31	CH15 delay hit
CINST_GATE_TYPE_NCOINS_S0	32	NCoins S0 flag

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
ci_set_tdc_clock_source(ci_handle_t handle, int32_t cs_type)	handle：对象句柄 cs_type: 时钟源类型	int - 状态码	设置 TDC 时钟源	详见 cinst_clock_source_t 定义
ci_set_data_mode(ci_handle_t handle, int32_t data_mode)	handle：对象句柄 data_mode: 数据模式	int - 状态码	设置数据模式	详见 cinst_data_mode_t 定义
ci_set_test_mode(ci_handle_t handle, int32_t test_mode)	handle：对象句柄 test_mode: 测试模式	int - 状态码	设置测试模式	详见 cinst_test_mode_t 定义

ci_set_test_data_bandwidth(ci_handle_t handle, int32_t bandwidth_type);	handle：对象句柄 bandwidth_type： 带宽类型	int - 状态码	设置测试数据带宽	详见 cinst_test_data_bandwidth_t 定义
ci_set_board_id(ci_handle_t handle, int32_t board_id);	handle：对象句柄 board_id: 板卡 ID	int - 状态码	设置新板卡 ID	范围：0~255
ci_set_coins_time_window(ci_handle_t handle, int32_t coins_value)	handle：对象句柄 coins_value: 时间窗口	int - 状态码	设置符合时间窗值	范围：0~160000000 ps
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 cinst_ad_area_scale_t 定义
ci_set_acq_type(ci_handle_t handle, int32_t acq_type)	handle：对象句柄 acq_type: 数据采集方式	int - 状态码	设置数据采集方式	详见 cinst_acq_type_t 定义
ci_set_output_gate_type(ci_handle_t handle, int32_t gate_type)	handle：对象句柄 gate_type: 输出门类型	int - 状态码	设置输出门类型	详见 cinst_output_gate_type_t 定义
ci_set_gate_delay(ci_handle_t handle, int32_t delay_value)	handle：对象句柄 delay_value: 延迟值	int - 状态码	设置门延迟值	
ci_set_single_countrate_window(ci_handle_t handle, int32_t time_ms)	handle：对象句柄 time_ms: 时间窗口值	int - 状态码	设置实时计数率窗口	范围：10ms-10s

ci_set_coins_countrate_window(ci_handle_t handle, int32_t time_ms)	handle：对象句柄 time_ms: 时间窗口值	int - 状态码	设置符合计数率窗口	范围：10ms-10s
ci_tdc_resetn(ci_handle_t handle)	handle：对象句柄	int - 状态码	TDC 复位	
ci_system_resetn(ci_handle_t handle)	handle：对象句柄	int - 状态码	系统复位	
ci_get_tdc_clock_source(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取 TDC 时钟源	
ci_get_data_mode(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取数据模式	
ci_get_test_mode(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取测试模式	
ci_get_test_data_bandwidth(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取测试数据带宽	
ci_get_board_id(ci_handle_t handle)	handle：对象句柄	int - 板卡 ID 号	获取板卡 ID	如果失败，会返回-1
ci_get_coins_time_window(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取符合时间窗值	
ci_get_ad_area_scale(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取 A/D 面积积分缩放类型	
ci_get_acq_type(ci_handle_	handle：对象句	int - 状态码	获取数据采集方	

t handle, int32_t* out_value)	柄 out_value:返回值指针		式	
ci_get_output_gate_type(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取输出门类型值	
ci_get_gate_delay(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取门延迟值	
ci_get_single_countrate_window(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取实时计数率窗口	范围：10ms-10s
ci_get_coins_countrate_window(ci_handle_t handle, int32_t* out_value)	handle：对象句柄 out_value:返回值指针	int - 状态码	获取符合计数率窗口	范围：10ms-10s

9.1.8 状态查询

【设备状态结构体-cinst_device_status_t】

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

current_7	float	电路 7 电流（mA）
current_8	float	电路 8 电流（mA）

【API 定义】

API 函数	参数	返回值	功能描述	注意事项
ci_get_channel_num(ci_handle_t handle)	handle: 对象句柄	int - 设备通道数	获取通道数量	获取设备通道数，不包含参考通道
ci_get_device_type(ci_handle_t handle)	handle: 对象句柄	int - 设备类型值	获取设备类型	0x1：Venus(TDC) 0x2: Mercury(多道分析仪) 0x3: Alea(真随机数生成器)
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/Alea-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0x10 - Venus Lite 4 0x11 - Venus Lite 2
ci_get_product_sn(ci_handle_t handle, char* out_value, int32_t size)	handle: 对象句柄 out_value :返回值指针 size：字符	int - 状态码	获取产品序列号	

	串大小			
ci_get_device_status(ci_handle_t handle, cinst_device_status_t* out_data)	handle: 对象句柄 out_data: 返回数据指针	int - 状态码	获取设备状态	包含温度和各电路电流，详见 cinst_device_status_t 定义
ci_get_device_temp(ci_handle_t handle, int32_t* out_temp)	handle: 对象句柄 out_temp: 返回设备核心温度	int - 状态码	获取设备核心温度	
ci_get_fan_speed(ci_handle_t handle)	handle: 对象句柄	int - 风扇转速	获取设备风扇转速（单位：RPM）	
ci_get_fw_version(ci_handle_t handle, char* out_value, int32_t size)	handle: 对象句柄 out_value :返回固件版本号字符串 size：字符串长度	int - 状态码	获取设备固件版本号	
ci_get_sw_version(ci_handle_t handle, char* out_value, int32_t size)	handle: 对象句柄 out_value :返回软件版本号字符串 size：字符串长度	int - 状态码	获取设备软件版本号	

9.1.9 网络配置

注意事项：

- 本机 IP 和设备 IP 需要在同一个网段；
- 注意本地端口的占用情况，请不要设置已被占用的端口号；

- 重新修改 IP 地址和网络端口后，请重连设备进行操作；
- 重新设置 MAC 地址后，需要重启网络。

【API 定义】

API 函数	参数	返回值	功能描述
ci_set_ips(ci_handle_t handle, const char* local_ip, const char* device_ip)	handle：对象句柄 local_ip：本地 IP device_ip：设备 IP	int - 状态码	设置千兆网口 IP 地址
ci_set_net_ports(ci_handle_t handle, int32_t local_port, int32_t device_port)	handle：对象句柄 local_port：本地端口 device_port：设备端口	int - 状态码	设置千兆网口端口号
ci_set_wips(ci_handle_t handle, const char* local_ip, const char* device_ip)	handle：对象句柄 local_ip：本地 IP device_ip：设备 IP	int - 状态码	设置万兆网口 IP 地址
ci_set_wnet_ports(ci_handle_t handle, int32_t local_port, int32_t device_port)	handle：对象句柄 local_port：本地端口 device_port：设备端口	int - 状态码	设置万兆网口端口号
ci_get_ips(ci_handle_t handle, char* out_local_ip, int32_t local_ip_size, char* out_device_ip, int32_t device_ip_size)	handle：对象句柄 out_local_ip:返回 Local IP 指针 local_ip_size:字符数组大小 out_device_ip:返回的 Device IP 指针 device_ip_size:字符数组大小	int - 状态码	获取千兆网口 IP 地址
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*	handle：对象句柄 out_local_ip:返回	int - 状态码	获取万兆网口 IP 地址

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

9.1.10 数据采集

注意事项：

- 数据采集 API 是异步的，API 调用后，不会阻塞线程，会立即返回，需要程序自己处理等待逻辑；
- API 返回失败，可以在错误回调中接收到具体的错误信息，或调用 get_last_error 方法查看错误代码和错误信息；
- 提供了实时获取当前数据的接口（请确保输出文件参数为空），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；
- 支持多种数据采集格式，可以根据实际需要，选择合适的格式类型；BIN 二进制格式数据最紧凑，数据采集效率最高。10G（万兆网）及以上高速数据采集场景，仅支持 BIN 二进制数据格式。

【数据采集格式定义-cinst_acq_data_format_t】

定义	值	说明
CINST_ACQ_DATAFORMAT_BIN	0	二进制格式
CINST_ACQ_DATAFORMAT_HEX	1	十六进制格式
CINST_ACQ_DATAFORMAT_DECODED	2	已解码格式

【API 定义】

API 函数	参数	返回值	说明
ci_set_acq_dataformat(ci_handle_t handle, int32_t data_format)	handle：对象句柄 data_foramt：数据采集格式类型	int - 状态码	设置数据采集格式
ci_get_acq_dataformat(ci_handle_t handle)	handle：对象句柄	int - 数据采集格式类型	获取数据采集格式
ci_acq_rawdata(ci_handle_t handle, int32_t data_mode, int32_t duration, const char* folder_path, const char* file_name, int32_t max_volumesize_mb, int8_t force_close)	handle：对象句柄 data_mode：数据模式 duration：数据采集时长 file_path：数据文件目录（不设置，默认是 raw 目录） file_name：数据文件名 max_volumesize_mb：文件分卷大小 force_close：是否	int - 状态码	开始进行数据采集（异步模式） <data_mode> 请详见前面的 DataMode 定义 <duration> 采集时间：单位-秒 <file_name> 未明确后缀名情况下： 二进制格式-默

	强制停止文件保存(1-强制停止；0-不强制停止，待内存数据都保存到文件)		认后缀名.bin; 十六进制-默认后缀名.dat; 已解码格式-默认后缀名.csv <max_volume size_mb> 如果分卷大小小于等于 0，文件不分卷
ci_fetch_rawdata(ci_handle_t handle, int32_t* out_size, int32_t timeout_ms)	handle：对象句柄 out_size:返回数组大小 timeout_ms：超时时间（单位：ms）	char*	实时读取 Raw 数据 返回 char 数组
ci_is_rawdata_processing(ci_handle_t handle, int8_t* out_value)	handle：对象句柄 out_value：返回布尔值	int - 状态码	是否在处理 RAW 数据
ci_stop_acq(ci_handle_t handle)	handle：对象句柄	int - 状态码	停止数据采集
ci_convert_to_hex(ci_handle_t handle, const char* bin_file, const char* hex_file)	handle：对象句柄 bin_file：二进制原始数据文件路径 hex_file:输出的十六进制文件路径	int - 状态码	转换 BIN 文件为 HEX 文件 如果 hex_file 输入为空，默认生成的十六进制文件在二进制文件同级目录下

9.1.11 数据分析

注意事项：

- 数据分析 API 是异步的，API 调用后，不会阻塞线程，会立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或调用 ci_get_last_error/ci_get_last_error_message 方法查看错误代码和错误信

息；

- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到了后，系统会自动调用 Stop API，停止分析。

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

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

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

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

【开始终止通道结构体-cinst_startstop_chan_t】

定义	类型	说明
start_chan	int32_t	开始通道号
stop_chan	int32_t	终止通道号

【NCoins 分析策略结构体-cinst_ncoins_strategy_t】

定义	类型	说明
strategy_name	const char*	策略名称
channels	int32_t*	符合通道号数组指针
chan_count	int32_t	符合通道号数组大小
virtual_twin	int32_t	虚拟时间窗

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

定义	值	说明

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

【FLIM 分析类型定义-cinst_flim_analysis_type_t】

定义	值	说明
CINST_FLIM_NONLOCATABLE_MODE	0	非定位模式
CINST_FLIM_LOCATABLE_MODE	1	定位模式

【FLIM 扫描类型定义-cinst_flim_scan_type_t】

定义	值	说明
CINST_FLIM_SCAN_RASTER	0	平行扫描
CINST_FLIM_SCAN_SERPENTINE	1	蛇形扫描

【API 定义】

API 函数	参数	返回值	功能描述	说明
ci_get_single_countrate(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	指定计数率统计窗口的计数率值
ci_get_coins_countrate(ci_handle_t handle, int32_t channel)	handle：对象句柄 channel:通道号	int-符合计数率的值	获取通道符合计数率	1）必须先设置 DATA MODE 为 Global Coins (全局符合模式) 2）指定计数

				率统计窗口的计数率值
ci_output_single_countrate(ci_handle_t handle, int32_t duration, const char* folder_path, const char* file_name)	handle：对象句柄 duration：数据采集时长 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_single_countrate_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_countrate(ci_handle_t handle, int32_t duration, const char* folder_path, const char* file_name)	handle：对象句柄 duration：数据采集时长 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_coins_countrate_recording(ci_handle_t handle)	handle：对象句柄	int-状态码	停止保存符合计数率	
ci_tof_analysis(ci_handle_t handle, int32_t duration, int32_t ch1, int32_t ch2, int32_t	handle：对象句柄 duration：分析时长 ch1：通道 1 通道号 ch2：通道 2 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

coins_time_window, int8_t enable_t3_data, int32_t avg_times, const char* folder_path, const char* file_name)	coins_time_window：符合时间窗值 enable_t3_data：是否保存 T3 数据 avg_times:开启平均触发模式后的平均次数 folder_path：文件夹路径 file_name：文件名			在 output 目录 <file_name> 如果不设置，不保存文件 <enable_t3_data> 如果启用，会保存 T3 数据到文件，或者缓存中待读取 <avg_times> 如果设置为 0，默认是单次触发模式
ci_set_cross_correlation_config(ci_handle_t handle, int32_t virtual_twin_ps, int32_t bin_width_ps, int32_t dcr1, int32_t dcr2)	handle：对象句柄 virtual_twin_ps：虚拟时间窗（单位：ps） bin_width_ps：BIN 宽度（单位：ps） dcr1：通道 1 的暗计数率（CPS） dcr2：通道 2 的暗计数率（CPS）	int-状态码	设置互关联分析参数	
ci_fetch_tof_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 Tof Process 数据	返回扁平 int64 数组，格式为：<tdiff(ps), counts>
ci_fetch_cross_correlation_data(ci_handle_t handle,	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不	double*	实时读取互关联分析数据	返回扁平 double 数组，格式为：

int32_t* out_data_points_size)	是数组长度）			<tdiff(ps), g2>
ci_stop_tof_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 TOF 分析	
ci_startstop_analysis(ci_handle_t handle, int32_t duration,const cinst_chan_group_t* chan_groups, int32_t chan_groups_count, const cinst_dual_stops_group_t* dual_stops_groups, int32_t dual_stops_groups_count, int32_t coins_time_window, int8_t enable_t3_data, const char* folder_path, const char* file_name)	handle：对象句柄 duration：分析时长 chan_groups：通道分组指针 chan_groups_count：通道分组数量 dual_stops_groups：双终止通道分组指针 dual_stops_groups_count:双终止通道分组数量 coins_time_window：符合时间窗值 enable_t3_data：是否保存 T3 数据 folder_path：文件夹路径 file_name：文件名	int-状态码	开始终止通道时间差分析	<chan_groups> cinst_chan_group_t 结构体数组指针 <dual_stops_groups> cinst_dual_stops_group_t 结构体数组指针 <enable_t3_data> 如果启用，会保存 T3 数据到文件，或者缓存中待读取
ci_fetch_startstop_processdata(ci_handle_t handle, int32_t start_chan, int32_t stop_chan, int32_t* out_data_points_size)	handle：对象句柄 start_chan：开始通道号 stop_chan：终止通道号 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 Start-stop Process 数据	返回扁平 int64 数组，格式为：<tdiff(ps), counts>
ci_fetch_startstop_average_processdata(ci_handle_t handle, int32_t start_chan, int32_t* out_data_points_size)	handle：对象句柄 start_chan：开始通道号 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 Start-stop Average Process 数据	返回扁平 int64 数组，格式为：<tdiff(ps), counts>
ci_fetch_startstop_dualstops_rawdata(ci_handle_t handle, int32_t start_chan,	handle：对象句柄 start_chan：开始通道号	int64_t*	实时读取 Start-stop dual stops	返回扁平 int64 数组，格式为：

int32_t* out_data_points_size, int32_t timeout_ms)	out_data_points_size:返回数值对的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）		RAW 数据	<x_stop_tdiff(ps), y_stop_tdiff(ps)>
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_apply_ncoins_strategies (ci_handle_t handle, const cinst_ncoins_strategy_t* ncoins_strats, int32_t ncoins_strats_count)	handle：对象句柄 ncoins_strats：分析策略组指针 ncoins_strats_count：分析策略组的大小	int-状态码	下发 N 重符合分析策略到设备（硬件分析模式）	<ncoins_strats> cinst_ncoins_strategy_t 结构体数组指针
ci_get_ncoins_countrate(ci_handle_t handle, int32_t strategy_index)	handle：对象句柄 strategy_index：分析策略组索引	uint32_t	实时读取指定策略的 N 重符合计数率	指定计数率统计窗口的计数率值
ci_ncoins_analysis(ci_handle_t handle, int32_t duration, const cinst_ncoins_strategy_t* ncoins_strats, int32_t ncoins_strats_count,int8_t enable_raw_data,const char* folder_path,const char* file_name)	handle：对象句柄 duration：数据采集时长 ncoins_strats：分析策略组指针 ncoins_strats_count：分析策略组的大小 enable_raw_data：是否记录 RAW 过程数据 folder_path：文件夹路径 file_name：文件名	int-状态码	N 重符合计数分析（软件分析模式）	<ncoins_strats> cinst_ncoins_strategy_t 结构体数组指针

ci_is_ncoins_analysis_processing(ci_handle_t handle, int8_t* out_value)	handle：对象句柄 out_value：返回布尔值	int - 状态码	是否在进行 N 重符合分析	可以通过本 API 去判断当前 N 重符合分析的状态，确保采集 duration 秒数据可以分析完成（N 重符合软件分析，依赖当前电脑硬件性能，分析时长会大于数据采集时长）
ci_fetch_ncoins_rawdata(ci_handle_t handle, int32_t strategy_index, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 strategy_index：分析策略组索引 out_data_points_size:返回数值组的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取 N 重符合计数分析的 RAW 过程数据	返回扁平 int64 数组，格式为：<coins_index, start_ch, start_timestamp(ps), end_ch, end_timestamp(ps)>
ci_fetch_ncoins_countrate(ci_handle_t handle, int32_t strategy_index, int32_t* out_data_points_size)	handle：对象句柄 strategy_index：分析策略组索引 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 N 重符合计数分析的计数率数据	1） 返回扁平 int64 数组，格式为：<index, count_rate> 2） 指定计数率统计窗口的计数率值
ci_stop_ncoins_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 N 重符合计数分析	

ci_flim_loc_analysis(ci_handle_t handle, int32_t duration, int32_t coins_time_window, int32_t frame_channel, int32_t line_channel, int32_t pixel_channel, int32_t start_channel, int32_t stop_channel, int32_t scan_type, const char* folder_path,const char* file_name)	handle：对象句柄 duration：分析时长 coins_time_window：符合时间窗值 frame_channel：帧通道 ID line_channel：行通道 ID pixel_channel：像素点通道 ID start_channel：开始通道 ID stop_channel：终止通道 ID scan_type：扫描模式 folder_path：文件夹路径 file_name：文件名	int-状态码	FLIM 定位模式分析	<scan_type> 扫描模式会拼接到输出文件名
ci_flim_nonloc_analysis(ci_handle_t handle, int32_t duration, int32_t coins_time_window, cinst_startstop_chan_t* startstop_chans,int32_t startstop_chans_count, const char* folder_path,const char* file_name)	handle：对象句柄 duration：分析时长 coins_time_window：符合时间窗值 startstop_chans：开始终止通道组 startstop_chans_count：开始终止通道组大小 folder_path：文件夹路径 file_name：文件名	int-状态码	FLIM 定位模式分析	

ci_stop_flim_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 FLIM 分析分析	
ci_autocorrelation_analysis(ci_handle_t handle, int32_t duration, int32_t ch, double min_tau, double max_tau, int bin_num, int8_t enable_t2_data, const char* folder_path, const char* file_name)	handle：对象句柄 duration：分析时长 ch：通道号 min_tau：最小 tau 值 max_tau：最大 tau 值 bin_num：BIN 数量 enable_t2_data：是否保存 T2 数据 folder_path：文件夹路径 file_name：文件名	int-状态码	单通道自关联函数（G2）分析	<enable_t2_data> 如果启用，会保存 T2 数据到文件，或者缓存中待读取
ci_fetch_autocorrelation_data(ci_handle_t handle, int32_t* out_data_points_size);	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	double*	实时读取自关联分析的分析数据	返回扁平 double 数组，格式为：<tdiff(s), g2>
ci_stop_autocorrelation_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止自关联函数分析	
ci_tot_analysis(ci_handle_t handle, int32_t duration, int32_t ch, int8_t enable_tot_data, const char* folder_path, const char* file_name)	handle：对象句柄 duration：分析时长 ch：通道 enable_tot_data：是否保存 TOT 脉宽数据 folder_path：文件夹路径 file_name：文件名	int-状态码	过阈时间脉宽分析	<enable_tot_data> 如果启用，会保存 TOT 脉宽数据到文件，或者缓存中待读取

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

				, counts>
ci_fetch_period_stability_metric(ci_handle_t handle, cinst_stability_metric_type_t metric_type, int32_t* out_data_points_size);	handle：对象句柄 metric_type：稳定性指标类型 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	double*	实时读取频率分析的稳定性指标数据	返回扁平 double 数组，格式为：<tau(s), value>
ci_stop_period_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止频率分析	
ci_ad_analysis(ci_handle_t handle, int32_t duration, int32_t ch, int32_t ad_integration_point, int32_t sampling_interval_ms, int32_t sampling_num, const char* folder_path, const char* file_name)	handle：对象句柄 duration：分析时长 ch：通道号 ad_integration_point: A/D 面积积分值 sampling_interval_ms：采样间隔时间 sampling_num：采样数量 folder_path：文件夹路径 file_name：文件名	int-状态码	A/D 信号分析	<ch> 必须是 A/D 通道 <sampling_interval_ms> 采样间隔，单位：ms <sampling_num> 如果小于等于 0，全采样
ci_fetch_ad_rawdata(ci_handle_t handle, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取 A/D RAW 数据	返回扁平 int64 数组，格式为：<sampling_points, adc>
ci_stop_ad_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止 A/D 信号分析	
ci_es_analysis(ci_handle_t handle, int32_t duration, int32_t ch, int32_t ad_integration_point, int32_t ad_area_scale,	handle：对象句柄 duration：分析时长 ch：通道 ad_integration_point: A/D	int-状态码	能谱分析	<ch> 必须是 A/D 通道 <enable_es_data>

int8_t enable_es_data, const char* folder_path, const char* file_name)	面积积分值 ad_area_scale: A/D 面积积分缩放类型 enable_es_data：是否保存 ES 能量数据 folder_path：文件夹路径 file_name：文件名			如果启用，会保存 ES 能量数据到文件，或者缓存中待读取
ci_fetch_es_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 ES 分析数据	返回扁平 int64 数组，格式为：<adc, counts>
ci_stop_es_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止能谱分析	
ci_conform_es_analysis(ci_handle_t handle, int32_t duration, int32_t ch1, int32_t ch2, int32_t ad_integration_point, int32_t ad_area_scale, int32_t energy_window_min1, int32_t energy_window_max1, int32_t energy_window_min2, int32_t energy_window_max2, int32_t coins_time_window, int32_t avg_times, const char* folder_path, const char* file_name)	handle：对象句柄 duration：分析时长 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 file_name：文件名	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
ci_fetch_conform_es_proc	handle：对象句柄	int64_t*	实时读取	返回扁平

essdata(ci_handle_t handle, int32_t* out_data_points_size)	out_data_points_size:返回数值对的大小（请注意：不是数组长度）		CONFORMES 分析数据	int64 数组，格式为：<tdiff(ps), counts>
ci_fetch_conform_es_area1_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 CONFORMES Area1 分析数据（低通道号）	返回扁平 int64 数组，格式为：<adc, counts>
ci_fetch_conform_es_area2_processdata(ci_handle_t handle, int32_t* out_data_points_size)	handle：对象句柄 out_data_points_size:返回数值对的大小（请注意：不是数组长度）	int64_t*	实时读取 CONFORMES Area2 分析数据（高通道号）	返回扁平 int64 数组，格式为：<adc, counts>
ci_stop_conform_es_analysis(ci_handle_t handle)	handle：对象句柄	int-状态码	停止符合能谱分析	
ci_fetch_t2_data(ci_handle_t handle, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 out_data_points_size:返回数值组的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取 T2 数据	返回扁平 int64 数组，格式为：<channel, timestamp(ps)>
ci_fetch_t3_data(ci_handle_t handle, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 out_data_points_size:返回数值组的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取 T3 数据	返回扁平 int64 数组，格式为：<index, ch1, ch2, tdiff(ps)>
ci_fetch_tot_data(ci_handle_t handle, int32_t*	handle：对象句柄 out_data_points_size:返回数值组的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取 TOT 脉宽数据	返回扁平 int64 数组，格式为：<channel, tot(ps)>

out_data_points_size, int32_t timeout_ms)				
ci_fetch_es_data(ci_handle_t handle, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 out_data_points_size:返回数值组的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取ES 能量数据	返回扁平int64 数组，格式为：<channel, area>
ci_fetch_flim_loc_data(ci_handle_t handle, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 out_data_points_size:返回数值组的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取FLIM 定位数据	返回扁平int64 数组，格式为：<frame_timestamp(ps), line_timestamp(ps), pixel_index, tdiff(ps)>
ci_fetch_flim_nonloc_data(ci_handle_t handle, int32_t* out_data_points_size, int32_t timeout_ms)	handle：对象句柄 out_data_points_size:返回数值组的大小（请注意：不是数组长度） timeout_ms：超时时间（单位：ms）	int64_t*	实时读取FLIM 非定位数据	返回扁平int64 数组，格式为：<timestamp(ps), start_chan, stop_chan, tdiff(ps)>
ci_wait_data_ready(ci_handle handle, int32_t timeout_ms)	handle：对象句柄 timeout_ms：超时时间（单位：ms）	int-状态码	确保数据分析准备完成	每次数据分析前，可以先调用这个API，确保数据采集通路准备完成（处理数据缓存残留等问题）
ci_clear_analysis_data(ci_handle_t handle)	handle：对象句柄	int-状态码	清除数据分析缓存	

9.1.12 其他辅助接口

【API 定义】

API 函数	参数	返回值	功能描述	注意事项
ci_free_char_data(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(int64_t* ptr)	ptr：内存指针	-	释放 API 分配的 Int64 内存指针	主要针对数据分析相关 API 返回的 Int64 内存指针，需要手动进行释放
ci_free_double_data(double* ptr)	ptr：内存指针	-	释放 API 分配的 Double 内存指针	主要针对数据分析相关 API 返回的 Double 内存指针，需要手动进行释放
ci_copy_data(char* dest, uint64_t src_addr, int size)	dest：目的内存指针 src_addr：API 分配的内存指针值	-	拷贝 API 分配的内存到目的内存地址	主要给 LabVIEW 调用
ci_copy_int64_data(int64_t* dest, uint64_t src_addr, int size)	dest：目的 Int64 内存指针 src_addr：API 分配的内存指针值	-	拷贝 API 分配的内存到目的内存地址	主要给 LabVIEW 调用
ci_copy_double_data(double* dest, uint64_t src_addr, int size)	dest：目的 double 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,	dest：目的内存指针 dest_size：目的内存大小 data：源数据地址 data_points_size	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。

const int64_t* data, int32_t data_points_size)	：源数据组大小			
ci_double_pairs_to_string(char* dest, int32_t dest_size, const double* data, int32_t data_points_size)	dest：目的内存指针 dest_size：目的内存大小 data：源数据地址 data_points_size：源数据组大小	int-实际字节数	转换 double 数据到字符串	Double 是数据分析 API 返回的一维数据，每 2 位数字表示一组 pair 数据。

9.1.13 通用注意事项

1) 状态码：

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

2) 线程安全：

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

9.2 C++ API

9.2.1 概述

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

操作的 C++封装类，提供设备连接、配置、数据采集和分析等功能。该类基于 Qt 框架开发，支持信号槽机制和异步操作。

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

9.2.2 初始化与设备连接

9.2.2.1 初始化

需要指定 API 配置文件路径，可以是绝对路径，也可以是相对路径。

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

【API 定义】

API	参数	参数说明	备注
std::shared_ptr<ChronosInst> create (const std::string& configPath, const std::string& 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
std::vector<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
bool connect(const std::string& deviceSN)	deviceSN	设备接口序列号	仅支持 USB3 接口类型

9.2.2.3 错误处理

【错误码定义】

定义	值	说明
ERROR_CONFIG_NOT_FOUND	2000	配置文件不存在
ERROR_INST_DEVICE_NOT_FOUND	2001	设备未找到
ERROR_INST_DEVICE_NOT_CONNECTED	2002	设备未连接
ERROR_INST_DEVICE_ALREADY_RUNNING	2003	设置已运行
ERROR_INST_START_PROTOCOL_FRAMEWORK	2004	启动协议层失败
ERROR_INST_PROTOCOL_FRAMEWORK_NOT_RUNNING	2005	协议层未运行
ERROR_INST_PROTOCOL_FRAMEWORK_INTERNAL	2006	协议层内部错误
ERROR_INST_INVALID_ACQ_TYPE	2007	错误的数据采集方式
ERROR_INST_ACQ_DATA_FAILED	2008	数据采集失败
ERROR_INST_PROCESS_ANALYSIS_FAILED	2009	数据分析失败
ERROR_INST_DATA_PROCESS_NOT_STOPPED	2010	上一次数据分析任务未完成
ERROR_INST_INVALID_CALIBRATION_PARAM	2011	错误的标定参数
ERROR_INST_CALIBRATION_FAILED	2012	设备标定失败
ERROR_INST_CONVERT_BINFILE_FAILED	2013	转换 BIN 文件失败
ERROR_INST_INVALID_PARAMETER	2014	错误的参数
ERROR_INST_TERMINATION_CONFIG_UNSUPPORTED	2015	不支持阻抗参数配置
ERROR_INST_ENABLE_CHANNEL_FAILED	2016	通道使能失败
ERROR_INST_DISABLE_CHANNEL_FAILED	2017	通道禁能失败
ERROR_INST_SET_CHANNELS_ENABLED_FAILED	2018	设置所有通道是否使能失败

【回调函数定义】

```
using ErrorCallback = std::function<void(INST_ERROR, const std::string&)>;
using StatusCallback = std::function<void(int)>;
```

【API 定义】

API	参数	参数说明	备注
void onErrorOccurred(ErrorCallback callback)	callback	INST_ERROR-错误码 descriptio-错误信息	请详见前面错误码定义
void onConvertStatus(StatusCallback callback)	callback	status-状态码	CONVERT_STATUS_FAIL = -1 CONVERT_STATUS_STOP = 0 CONVERT_STATUS_SUCCESS = 1

9.2.3 设备自检

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

【API 定义】

API	参数	参数说明	备注
bool selfCheck()	-	-	设备自检接口 返回成功或失败

9.2.4 通道配置

9.2.4.1 DAC 设置

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

注意事项：

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

【API 定义】

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

9.2.4.2 阻抗设置

可以单独设置各个通道的阻抗值（有效值：50 Ohm 和 1M Ohm）。

注意事项：

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

【阻抗类型（INST_TERM_TYPE）】

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

【API 定义】

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

9.2.4.3 迟滞电压设置

设置通道的迟滞电压类型值（有效值范围：0-3，超出会失败）。

注意事项：

- 参考通道不支持设置迟滞电压。

【通道迟滞电压类型（INST_HTS_TYPE）】

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

【API 定义】

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

9.2.4.4 通道间延迟设置

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

【API 定义】

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

9.2.4.5 死时间设置

设置各个通道的死时间值（有效值范围：2-130000ps，超出会失败）。

【API 定义】

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

9.2.4.6 A/D 面积积分设置

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

注意事项：

- 参考通道不支持设置 A/D 面积积分值。

【API 定义】

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

9.2.4.7 边沿触发类型设置

【通道边沿触发类型】

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

【API 定义】

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

9.2.4.8 通道使能设置

可以对通道设置使能/禁能。禁能后，该通道关闭，不再输出测量数据。

【API 定义】

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

9.2.5 全局配置

9.2.5.1 板卡 ID 设置

设置当前设备板块的 ID 号，主要用于设备并联时区分设备使用（有效值范围：

0-255，超出会失败）。

【API 定义】

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

9.2.5.2 TDC 时钟来源设置

【TDC 时钟来源】切换后，设备需要重新校准，耗时较长，一般需要 30 秒左右

定义	值	说明
INST_INTERNAL_CLOCK	0	本地时钟
INST_EXTERNAL_CLOCK_25MHZ	1	外部时钟（25MHz）
INST_EXTERNAL_CLOCK_10MHZ	2	外部时钟（10MHz）

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setTdcClockSource(int srcType)	srcType	时钟来源类型	设置和获取 TDC 时钟来源 返回 INST_RESULT<int> status - 状态 value - 时钟来源类型值
INST_RESULT<int> getTdcClockSource()	-	-	

9.2.5.3 数据模式设置

【数据模式】

定义	值	说明
INST_DATA_MODE_TIMESTAMP	0	时间模式
INST_DATA_MODE_TIME ZONE	1	时区模式
INST_DATA_MODE_AD	2	A/D 模式
INST_DATA_MODE_AREA	3	面积测量模式
INST_DATA_MODE_REF_COINS	4	参考符合模式
INST_DATA_MODE_GLOBAL_COINS	5	全局符合模式
INST_DATA_MODE_TOT	6	过阈时间测量模式
INST_DATA_MODE_AREA_COINS	7	能谱-全局符合模式
INST_DATA_MODE_DUAL_TIMESTAMP	8	双沿时间模式
INST_DATA_MODE_PERIOD	9	周期测量模式
INST_DATA_MODE_SINGLE_COINS	11	单符合模式
INST_DATA_MODE_GLOBAL_COINS_TIMESTAMP	12	全局符合（带时间戳）模式
INST_DATA_MODE_SINGLE_TIMESTAMP	13	单边沿时间戳模式
INST_DATA_MODE_GLOBAL_COINS2	14	全局符合模式 2

【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

9.2.5.4 符合时间窗设置

设置符合时间窗数值（有效值范围：0-16000000ps，超出会失败）。

【API 定义】

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

9.2.5.5 测试模式设置

【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_INTERNAL_TRIGGER	6	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

INST_RESULT<int> getTestMode()	-	-	返回 INST_RESULT<int> status - 状态 value - 测试模式类型值

9.2.5.6 数据带宽设置

【数据带宽】

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

【API 定义】

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

9.2.5.7 A/D 面积积分缩放类型设置

【A/D 面积积分缩放类型】

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

INST_AREA_SCALE_1_128	7	1/128 倍

【API 定义】

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

9.2.5.8 数据采集方式设置

【数据采集方式】

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

【API 定义】

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

9.2.5.9 输出门类型和延迟设置

【输出门类型】

定义	值	说明
INST_GATE_TYPE_ANY_COINS	0	Any coincidence flag
INST_GATE_TYPE_CH0_CH1_COINS	1	CH0 and CH1 coincidence flag
INST_GATE_TYPE_CH2_CH3_COINS	2	CH2 and CH3 coincidence flag
INST_GATE_TYPE_CH4_CH5_COINS	3	CH4 and CH5 coincidence flag

INST_GATE_TYPE_CH6_CH7_COINS	4	CH6 and CH7 coincidence flag
INST_GATE_TYPE_CH8_CH9_COINS	5	CH8 and CH9 coincidence flag
INST_GATE_TYPE_CH10_CH11_COINS	6	CH10 and CH11 coincidence flag
INST_GATE_TYPE_CH12_CH13_COINS	7	CH12 and CH13 coincidence flag
INST_GATE_TYPE_CH14_CH15_COINS	8	CH14 and CH15 coincidence flag
INST_GATE_TYPE_ANY_HIT	9	Any hit flag
INST_GATE_TYPE_CH0_HIT	10	CH0 hit flag
INST_GATE_TYPE_CH1_HIT	11	CH1 hit flag
INST_GATE_TYPE_CH2_HIT	12	CH2 hit flag
INST_GATE_TYPE_CH3_HIT	13	CH3 hit flag
INST_GATE_TYPE_SYSTEM_CLOCK_PLL_LOCKED	14	System clock PLL locked
INST_GATE_TYPE_I_GATE	15	i_gate
INST_GATE_TYPE_CH0_DELAY_HIT	16	CH0 delay hit
INST_GATE_TYPE_CH1_DELAY_HIT	17	CH1 delay hit
INST_GATE_TYPE_CH2_DELAY_HIT	18	CH2 delay hit
INST_GATE_TYPE_CH3_DELAY_HIT	19	CH3 delay hit
INST_GATE_TYPE_CH4_DELAY_HIT	20	CH4 delay hit
INST_GATE_TYPE_CH5_DELAY_HIT	21	CH5 delay hit
INST_GATE_TYPE_CH6_DELAY_HIT	22	CH6 delay hit
INST_GATE_TYPE_CH7_DELAY_HIT	23	CH7 delay hit
INST_GATE_TYPE_CH8_DELAY_HIT	24	CH8 delay hit
INST_GATE_TYPE_CH9_DELAY_HIT	25	CH9 delay hit
INST_GATE_TYPE_CH10_DELAY_HIT	26	CH10 delay hit
INST_GATE_TYPE_CH11_DELAY_HIT	27	CH11 delay hit
INST_GATE_TYPE_CH12_DELAY_HIT	28	CH12 delay hit
INST_GATE_TYPE_CH13_DELAY_HIT	29	CH13 delay hit
INST_GATE_TYPE_CH14_DELAY_HIT	30	CH14 delay hit
INST_GATE_TYPE_CH15_DELAY_HIT	31	CH15 delay hit

INST_GATE_TYPE_NCOINS_S0	32	NCoins S0 flag

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setOutputGateType(int gateType)	acqType	设置输出门类型	设置和获取输出门类型 返回 INST_RESULT<int> status - 状态 value - 输出门类型值
INST_RESULT<int> getOutputGateType()	-	获取输出门类型	
INST_RESULT<int> setGateDelay(int delay_value)	delay_value	设置门延迟值	设置和获取门延迟值 返回 INST_RESULT<int> status - 状态 value - 门延迟值
INST_RESULT<int> getGateDelay()	-	获取门延迟值	

9.2.5.10 计数率窗口设置

【API 定义】

API	参数	参数说明	备注
INST_RESULT<int> setSingleCountRateWindow(int time_ms)	timeMs	设置实时计数率统计窗口值	范围：10ms-10000ms
INST_RESULT<int> getSingleCountRateWindow()	-	获取实时计数率统计窗口值	
INST_RESULT<int> setCoinsCountRateWindow(int time_ms)	timeMs	设置符合计数率统计窗口值	范围：10ms-10000ms
INST_RESULT<int> getCoinsCountRateWindow()	-	获取实时计数率统计窗口值	

9.2.5.11 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） 0x3: Alea（真随机数生成器）
int getDeviceMode()	获取设备型号	0x1: Venus Lite-32/Mercury-16/Alea-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0x10 - Venus Lite 4 0x11 - Venus Lite 2
INST_RESULT<std::string> getProductSN()	获取设备产品序列号	INST_RESULT<std::string> status - 状态 value - 面积积分缩放类型值

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

int getFanSpeed()	获取风扇转速（单位:	

	RPM）	
const std::string getFWVersion()	获取固件版本号	
const std::string getSWVersion()	获取软件版本号	

9.2.5.11 复位控制

主要包括 TDC 复位和系统全局复位。

注意事项：

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

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

9.2.6 网络配置

9.2.6.1 千兆网 IP 地址设置

设置千兆网口的 IP 地址。

注意事项：

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

【API 定义】

API	参数	参数说明	备注
INST_NET_RESULT<std::string> setIPs(const std::string& local_ip,	local_ip	本地 IP 地址	设置和获取千兆网 IP 地址 返回

const std::string& device_ip)	device_ip	设备 IP 地址	INST_NET_RESULT<std::string> status - 状态 local_value - 本地 IP 地址 device_value - 设备 IP 地址
INST_NET_RESULT<std::string> getIPs()	-	-	

9.2.6.2 千兆网端口设置

设置千兆网口的网络端口。

注意事项：

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

【API 定义】

API	参数	参数说明	备注
INST_NET_RESULT<int> setNetPorts(int local_port, int device_port)	local_port	本地网络端口	设置和获取千兆网端口 返回 INST_NET_RESULT<int> 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()	-	-	

9.2.6.3 万兆网 IP 设置

设置万兆网口的 IP 地址。

注意事项：

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

【API 定义】

API	参数	参数说明	备注

INST_NET_RESULT<std::string> setWIPs(const std::string& local_ip, const std::string& device_ip)	local_ip	本地 IP 地址	设置和获取万兆网 IP 返回 INST_NET_RESULT<std::string> 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<std::string> getWIPs()	-	-	

9.2.6.4 万兆网端口设置

设置万兆网口的网络端口。

注意事项：

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

【API 定义】

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

9.2.6.5 千兆网 MAC 地址设置

设置设备千兆网口 MAC 地址。

注意事项：

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

【API 定义】

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

9.2.6.6 万兆网 MAC 地址设置

设置设备万兆网口的 MAC 地址。

注意事项：

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

【API 定义】

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

9.2.7 数据采集

指定数据模式下的 RAW 数据采集接口。

注意事项：

- 数据采集 API 是异步的，API 调用后，不会阻塞线程，会立即返回，需要程序自己处理等待逻辑；

- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 提供了实时获取当前数据的接口（请确保输出文件参数为空），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；
- 支持多种数据采集格式，可以根据实际需要，选择合适的格式类型；BIN 二进制格式数据最紧凑，数据采集效率最高。10G（万兆网）及以上高速数据采集场景，仅支持 BIN 二进制数据格式。

【数据采集格式类型】

定义	值	说明
INST_ACQ_DATAFORMAT_BIN	0	二进制格式
INST_ACQ_DATAFORMAT_HEX	1	十六进制格式
INST_ACQ_DATAFORMAT_DECODED	2	已解码格式

【API 定义】

API	参数	参数说明	备注
bool setAcqDataFormat(int dataFormat)	dataFormat	数据采集格式类型（详见前面数据采集格式定义）	设置数据采集格式
int getAcqDataFormat()	-	-	返回数据采集格式类型
bool acqRawData(int dataMode, int duration，const std::string& folderPath = ""	dataMode	数据模式（详见前面的数据模式定义）	开始数据采集（异步模式，立即返回） 返回成功或失败。 如果设置了输出文件名，会启用数据
	duration	数据采集时长（单位：秒）	
	folderPath	输出文件目录设置（默认是 raw 目录）	

, const std::string& fileName = "", int maxVolumeSizeMB = -1, bool forceClose = true)	fileName	输出文件名称（如果不指定，不保存文件）	文件保存。未明确后缀名情况下：二进制格式-默认后缀名.bin；十六进制-默认后缀名.dat；已解码格式-默认后缀名.csv
	maxVolumeSizeMB	文件分卷大小（默认不分卷，单位：MB）	
	forceClose	是否强制停止文件保存（默认是true，如果需要等待内存中数据全部保存，请设置为false）	
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 isRawDataProcessing()	-	-	是否在处理 RAW 数据
bool stopAcq()	-	-	停止数据采集 返回成功或失败
bool convertToHex(const std::string& binFile, const std::string& hexFile, bool isAsync = false)	binFile	二进制原始数据文件路径	转换二进制格式文件到十六进制格式（请确认目标文件是二进制文件格式）
	hexFile	输出的十六进制文件路径	如果输入为空，默认生成的十六进制文件在二进制文件同级目录下
	isAsync	是否异步处理	如果异步处理，方法立即返回；需要调用方侦听转换状态信号
void stopAsyncConvert()	-	-	如果采用异步处理模式，可以调用停止方法，停止转换

			操作

9.2.8 数据分析

9.2.8.1 强度分析

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

注意事项：

- 文件输出 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的任务；如果不调用，时间到了后，系统会自动调用 Stop API，停止任务；
- 输出计数率前，请确认计数率统计窗口的设置值（默认是 1s）；
- 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 outputSingleCountRate(int duration,const std::string& folderPath, const std::string& 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 stopSingleCountRateRecording()	-	-	停止实时计数率保存任务 返回成功或失败
bool outputCoinsCountRate(int	duration	数据采集时间（单	输出各通道符合计

duration,const std::string& folderPath, const std::string& fileName)		位：秒）	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 stopCoinsCountRateRecording()	-	-	停止符合计数率保存任务 返回成功或失败
int getSingleCountRate(int channel)	channel	通道号	获取指定通道的实时计数率
int getCoinsCountRate(int channel)	channel	通道号	获取指定通道的符合计数率

9.2.8.2 TOF 分析（双通道时间差分析）

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

注意事项：

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

【2D 数据点结构体-Point2D】

字段	参数类型	参数说明
x	double	二维点的 X 坐标
y	double	二维点的 Y 坐标

【API 定义】

API	参数	参数说明	备注
bool tofAnalysis(int duration, const INST_TOF_OPTIONS& options, const std::string& folderPath = "", const std::string& fileName = "")	duration	数据分析时长	单位：秒
	options	INST_TOF_OPTIONS { ch1, ch2, coins_time_window, enable_avg_mode, avg_times, enable_t3_data }	<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 <enable_t3_data> 如果启用， 会保存 T3 数据到文件，或者缓存中待读取
	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 setCrossCorrelationConfig(const INST_CROSS_CORR_CFG& cfg)	cfg	设置互关联分析参数	返回成功或失败
bool fetchT3Data(std::vector<INST_T3_DATA>& outData, int timeout_ms)	outData	返回 T3 数据	INST_T3_DATA { uint32_t index = 0; int ch1 = -1; int ch2 = -1; uint32_t tdiff = 0;

			};
	timeout_ms	获取数据超时设置（ms）	
bool fetchTofProcessData(std::vector<csp::Point2D>& outData)	outData	返回 TOF 分析数据	Point2D 格式 X-tdiff 值 Y-counts 数量
bool fetchCrossCorrelationData(std::vector<csp::Point2D>& outData)	outData	返回互关联分析数据	Point2D 格式 X-tdiff 值 Y-g2 值
bool stopTofAnalysis()	-	-	停止 TOF 分析 返回成功或失败

9.2.8.3 开始-终止通道时间差分析

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

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 对于双终止通道分析，必须通道分组中，终止通道多于 2 个，并且指定不同的 X-终止通道和 Y-终止通道；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool startStopAnalysis(int duration, const	duration	数据分析时长	单位：秒
	options	INST_START_STOP_OP	<chan_groups>

INST_START_STOP_OPTIONS& options, const std::string& folderPath = "", const std::string& fileName = "")		TIONS{ chan_groups, dual_stops_groups, coins_time_window, enable_t3_data}	开始终止分组列表 <dual_stops_groups> 双终止通道分组列表 <coins_time_window> 符合时间窗值 <enable_t3_data> 如果启用， 会保存 T3 数据到文件，或者缓存中待读取
	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
fetchT3Data(std::vector<INST_T3
_DATA>& outData, int
timeout_ms)
	outData	返回 T3 数据	INST_T3_DATA {
			uint32_t index = 0;
			int ch1 = -1;
			int ch2 = -1;
			uint32_t tdiff = 0;
			};

	timeout_ms	获取数据超时设置（ms）	
bool fetchStartStopProcessData(int start_chan, int stop_chan, std::vector<csp::Point2D>& outData)	start_chan	开始通道号	
	stop_chan	终止通道号	
	outData	返回 Start-stop Process 数据	Point2D 格式 X-tdiff 值 Y-counts 数量
bool fetchStartStopAverageProcessData(int start_chan,	start_chan	开始通道号	
	outData	返回 Start-stop	Point2D 格式

std::vector<csp::Point2D>& outData)		Average Process 数据	X-tdiff 值 Y-counts 数量
bool fetchStartStopDualStopsRawData(int start_chan, std::vector<csp::Point2D>& outData, int timeout_ms)	start_chan	开始通道号	
	outData	返回 Start-stop Dual Stops Raw 数据	Point2D 格式 X-stop tdiff 值 Y-stop tdiff 值
	timeout_ms	超时时间（单位：ms）	
bool stopStartStopAnalysis()	-	-	停止开始终止分析 返回成功或失败

9.2.8.4 N 重符合计数分析

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

注意事项：

- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；
- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析；
- 调用前，请确认和设置计数率统计窗口时间值（范围：10ms-10s）。

【API 定义】

API	参数	参数说明	备注
bool applyNCoinsStrategies(const QVector<INST_NCOINS_STRATEGY>& ncoinsStrats)	ncoinsStrats	N 重符合分析的策略组	下发 N 重符合分析策略组到设备（硬件分析模式）

uint32_t getNCoinsCountRate(int strategy_index)	strategy_index	策略索引	返回指定策略的符合计数率（指定计数率统计窗口时间）
bool nCoinsAnalysis(int duration, const INST_NCOINS_OPTIONS& options, const std::string& folderPath = "", const std::string& fileName = "")	duration	数据采集时长	单位：秒

	options	INST_NCOINS_OPTIONS{
		ncoins_strats;
		enable_raw_data;}
		
		INST_NCOINS_STRATEGY{
		strategy_name;
		channels;
		virtual_twin;
		}
			N 重符合计数分析策略配置
			<ncoins_strats>
			N 重符合分析的策略组
			<enable_raw_data>
			如果启用， 会保存 RAW 数据到文件，或者缓存中待读取
			
			N 重符合计数分析策略
			<strategy_name>
			分析策略名称
			<channels>
			通道号组
			<virtual_twin>
			虚拟时间窗

	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为 “.csv”
bool isNCoinsAnalysisProcessing()	-	-	可以通过本 API 去判断 当前 N 重符 合分析的状态，确保采 集 duration 秒数据可以分析完成（N 重符合软件分析，依赖当前电脑硬件性能，分析时长会大于数据采集时长）
bool fetchNCoinsRawData(int stratIndex,	stratIndex	策略索引	
	outData	返回指定策略的符合计数数据	INST_NCOINS_DATA{ coins_index; (当前符合的 Index) start_ch;

std::vector<INST_NCOINS_DATA>& outData, int timeout_ms)			(开始通道号) start_timestamp; (开始通道的时间戳) end_ch; （终止通道号） end_timestamp; （终止通道的时间戳） }
	timeout_ms	超时时间（单位：ms）	
bool fetchNCoinsCountRate(int stratIndex, std::vector<csp::Point2D>& outData)	stratIndex	策略索引	
	outData	返回指定策略的计数率数据	Point2D 格式 X-index Y-count rate
bool stopNCoinsAnalysis()	-	-	停止 N 重符合分析 返回成功或失败

9.2.8.5 FLIM 分析

FLIM 分析，数据以文件格式输出，也可以实时从内存读取分析数据。

注意事项：

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

【FLIM 分析类型定义-INST_FLIM_ANALYSIS_TYPE】

定义	值	说明
INST_FLIM_NONLOCATABLE_MODE	0	非定位模式
INST_FLIM_LOCATABLE_MODE	1	定位模式

【FLIM 扫描类型定义-INST_FLIM_SCAN_TYPE】

定义	值	说明
CINST_FLIM_SCAN_RASTER	0	平行扫描
CINST_FLIM_SCAN_SERPENTINE	1	蛇形扫描

【API 定义】

API	参数	参数说明	备注
bool flimAnalysis(int duration, const INST_FLIM_OPTIONS& options, const std::string& folderPath = "", const std::string& fileName = "")	duration	数据分析时长	单位：秒
	options	INST_FLIM_OPTIONS { int analysis_mode; int frame_chan; int line_chan; int pixel_chan; int start_chan; int stop_chan; int scan_type; QSet<QPair<int, int>> startstop_groups;	FLIM 分析配置 <analysis_mode> 分析模式：非定位模式、定位模式 <frame_chan> 帧通道 ID <line_chan> 行通道 ID <pixel_chan> 图像像素通道 ID <start_chan> 开始通道 ID <stop_chan> 终止通道 ID <scan_type> 扫描方式：平行扫描或蛇形扫描 <startstop_groups> 开始终止分组列表

		int coins_time_window; };	<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 fetchFlimLocData(std::vector<INST_FLIM_LOC_DATA>& outData, int timeout_ms)	outData	返回 FLIM 定位数据	INST_FLIM_LOC_DATA{ frame_timestamp; (帧通道时间戳) line_timestamp; (行通道时间戳) pixel_index; (像素 index) tdiff; （时间差值） }
	timeout_ms	超时时间（单位：ms）	
bool fetchFlimNonLocData(std::vector<INST_FLIM_NONLOC_DATA>& outData, int timeout_ms)	outData	返回 FLIM 非定位数据	INST_FLIM_NONLOC_DATA{ timestamp; (时间戳) start_chan; (开始通道号) stop_chan; (终止通道号) tdiff; （时间差值） }
	timeout_ms	超时时间（单位：	

		ms）	
bool stopFlimAnalysis()	-	-	停止 FLIM 分析 返回成功或失败

9.2.8.6 单通道自关联函数分析

对指定通道进行自关联函数（G2）分析，数据以文件格式输出，也可以实时从内存读取分析数据。

注意事项：

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

【API 定义】

API	参数	参数说明	备注
bool autoCorrelationAnalysis(int duration, const INST_AUTO_CORRELATION_OPTI ONS& options, const std::string& folderPath = "", const std::string& fileName = "")	duration	数据分析时长	单位：秒

	options	INST_AUTO_CORRELATION_OPTIONS{
		ch;
		min_tau;
		max_tau;
		bin_num;
		enable_t2_data;
		}
			自关联函数分析配置
			<ch>
			分析的通道号
			<min_tau>
			最小 tau 值（单位：s）
			<max_tau>
			最大 tau 值（单位：s）
			<bin_num>
			BIN 数量
			<enable_t2_data>
			如果启用，会保存 T2 数据到文件，或者缓存中待读取

	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 fetchT2Data(std::vector<INST_T2_DATA>& outData, int timeout_ms)	outData	返回 T2 数据	INST_T2_DATA{
			channel;
			(通道号)
			timestamp;
			(触发时间戳)
			}

	timeout_ms	超时时间（单位：ms）	
bool fetchAutoCorrelationData(std::vector<csp::Point2D>& outData)	outData	返回自关联函数分析数据	Point2D 格式 X-tau（s） Y-g2 值
bool stopAutoCorrelationAnalysis()	-	-	停止自关联分析 返回成功或失败

9.2.8.7 TOT 分析（过阈时间分析）

对指定通道的 TOT（过阈时间分析）分析，数据以文件格式输出。

注意事项：

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

【API 定义】

API	参数	参数说明	备注
bool totAnalysis(int duration, const INST_TOT_OPTIONS&	duration	数据分析时长	单位：秒
	options	INST_TOT_OPTION	TOT 分析配置

options, const std::string& folderPath = "", const std::string& fileName = "")		S{
		channel;
		enable_tot_data;
		}
			<channel>
			分析的通道号
			<enable_tot_data>
			如果启用，会保存 TOT 脉宽数据到文件，或者缓存中待读取

	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
fetchTotData(std::vector<INST_TOT_DATA>& outData, int
timeout_ms)
	outData	返回 TOT 脉宽数据	INST_TOT_DATA{
			channel;
			(通道号)
			tot;
			(TOT 脉宽数据)
			}

	timeout_ms	超时时间（单位：ms）	
bool fetchTotProcessData(std::vector<csp::Point2D>& outData)	outData	返回 TOT 分析数据	Point2D 格式 X-tot 值 Y-counts 数量
bool stopTotAnalysis()	-	-	停止 TOT 分析 返回成功或失败

9.2.8.8 频率分析

对通道接入的信号进行频率分析，数据以文件格式输出。

注意事项：

- 请确保只接入 1 路外部信号（只有前 2 个通道可以进行频率分析）；
- 分析 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 如果 API 返回失败，可以在错误回调中接收到具体的错误信息，或在 log 日志中进行查看；

- 可以调用 Stop API 去强制停止当前的分析任务；如果不调用，分析时间到后，系统会自动调用 Stop API，停止分析。

【API 定义】

API	参数	参数说明	备注
bool periodAnalysis(int duration, const INST_PERIOD_OPTIONS& options, const std::string& folderPath = "", const std::string& fileName = "")	duration	数据分析时长	单位：秒

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

	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool fetchPeriodRawData(std::vector<csp::Point2D>& outData, int timeout_ms)	outData	返回抽样周期数据	Point2D 格式 X-index Y-period 周期值(单

			位：ps)
	timeout_ms	超时时间（单位：ms）	
bool fetchPeriodProcessData(std::vector<csp::Point2D>& outData)	outData	返回频率分析周期统计数据	Point2D 格式 X-period 值 Y-counts 数量

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

bool fetchPeriodStabilityMetric(INST_STABILITY_ METRIC_TYPE metricType , std::vector<csp::Point2D>& outData)	metricType	稳定性分析指标类型	
	outData	返回指定类型的稳定性分析数据	Point2D 格式 X-tau(s) Y-value
bool stopPeriodAnalysis()	-	-	停止频率分析 返回成功或失败

9.2.8.9 A/D 分析

对指定的通道进行 A/D 信号数据分析，数据以文件格式输出。

注意事项：

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

【API 定义】

API	参数	参数说明	备注
bool adAnalysis(int duration, const INST_AD_OPTIONS& options, const std::string& folderPath = "", const std::string& fileName = "")	duration	数据分析时长	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(std::vector<csp::Point2D>& outData, int timeout_ms)	outData	返回 A/D Raw 数据	Point2D 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()	-	-	停止 A/D 分析 返回成功或失败

9.2.8.10 能谱分析

对指定通道进行能谱信号数据分析，数据以文件格式输出。

注意事项：

- 该接口仅对 A/D 通道有效（设备奇数通道），请确保外接信号；

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

【API 定义】

API	参数	参数说明	备注
bool esAnalysis(int duration, const INST_ES_OPTIONS& options, const std::string& folderPath = "", const std::string& fileName = "")	duration	数据分析时长	单位：秒

	options	INST_ES_OPTIONS{
		ch;
		ad_integration_point;
		ad_area_scale;
		enable_es_data;
		}
			<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
			<enable_es_data>
			如果启用，会保存 ES 能量数据到文件，或者缓存中待读取

	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(std::vector<csp::Point2D>& outData)	outData	返回 ES Process 数据	Point2D 格式 X-adc 值 Y-counts 数量
bool stopEsAnalysis()	-	-	停止 ES 分析 返回成功或失败

9.2.8.11 符合能谱分析

对指定的两个 A/D 通道进行时间符合能谱数据分析，数据以文件格式输出。

注意事项：

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

【API 定义】

API	参数	参数说明	备注
bool conformEsAnalysis(int duration, const INST_CONFORM_ES_OPTIONS& options, const std::string& folderPath = "", const std::string& fileName = "")	duration	数据分析时长	单位：秒

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

			通道 2 最小能量窗设置 <energy_window_max2> 通道 2 最大能量窗设置 <coins_time_window> 符合时间窗值 <enable_avg_mode> 是否启用平均触发模式 <avg_times> 启用平均触发模式后，平均次数设置
	folderPath	数据文件输出目录	可设置绝对路径或相对路径。不设置，默认是 output 目录
	fileName	数据文件名称	未明确后缀名情况下，默认后缀名为“.csv”
bool fetchConformEsProcessData(std::vector<csp::Point2D>& outData)	outData	返回 Conform ES Process 数据	Point2D 格式 X-tdiff 值 Y-counts 数量
bool fetchConformArea1ProcessData(std::vector<csp::Point2D>& outData)	outData	返回 Conform ES Area1 Process 数据（对应低通道号）	Point2D 格式 X-adc 值 Y-counts 数量
bool fetchConformArea2ProcessData(std::vector<csp::Point2D>& outData)	outData	返回 Conform ES Area2 Process 数据（对应高通道号）	Point2D 格式 X-adc 值 Y-counts 数量
bool stopConformEsAnalysis()	-	-	停止 ConformES 返回成功或失败

9.2.8.12 数据分析辅助 API

其他辅助 API 定义。

【API 定义】

API	参数	参数说明	备注
void waitDataReady(int timeoutMs)	timeoutMs	超时时间（单位：ms）	每次数据分析前，可以先调用这个 API，确保数据采集通路准备完成（处理数据缓存残留等问题）
bool clearAnalysisData()	-	-	清除数据分析缓存

9.2.9 注意事项

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

9.2.10 示例代码

典型的应用代码示例如下：

```
int main(int argc, char *argv[])
{
    // 1. 初始化 TDC
    ChronosInst tdc =   ChronosInst::create("apiconfig.ini", "D:\\cinstapi");

    // 2. 设置错误回调
```

```
tdc.onErrorOccurred([](INST_ERROR error, const std::string& desc) {
    std::cout << static_cast<int>(error) << ": " << desc << std::endl;
});

// 3. 获取并连接设备
auto devices = tdc.getDevices();
if(!devices.empty()) {
    tdc.connect(devices[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.setAcqDataFormat(INST_ACQ_DATAFORMAT_HEX);

// 采集原始数据
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
std::this_thread::sleep_for(std::chrono::seconds(10));

// 停止采集
flag = tdc.stopAcq();

...
tdc.disconnect();
return 0;
```

```
}
```

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

【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	INST 实例	初始化 ChronosInst 实例	

9.3.3 设备连接

注意事项:

- 可以先调用 get_devices 获取设备接口列表后，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]	获取可用设备接口列表	设备信息包含类型、名称和设备接口序列号等
connect	ConnType device_type：设备接口类型	bool	连接设备	具体类型定义请详见 ConnType 定义。
connect_by_sn	device_sn：设备接口序列号	bool	连接设备（指定设备接口序列号）	仅支持 USB3 接口类型
disconnect	-	bool	断开当前设备连接	

9.3.4 错误处理

【错误码定义】

定义	值	说明
Error.CONFIG_NOT_FOUND	2000	配置文件不存在
Error.INST_DEVICE_NOT_FOUND	2001	设备未找到
Error.INST_DEVICE_NOT_CONNECTED	2002	设备未连接
Error.INST_DEVICE_ALREADY_RUNNING	2003	设置已运行
Error.INST_START_PROTOCOL_FRAMEWORK	2004	启动协议层失败
Error.INST_PROTOCOL_FRAMEWORK_NOT_RUNNING	2005	协议层未运行

Error.INST_PROTOCOL_FRAMEWORK_INTERNAL	2006	协议层内部错误
Error.INST_INVALID_ACQ_TYPE	2007	错误的数据采集方式
Error.INST_ACQ_DATA_FAILED	2008	数据采集失败
Error.INST_PROCESS_ANALYSIS_FAILED	2009	数据分析失败
Error.INST_DATA_PROCESS_NOT_STOPPED	2010	上一次数据分析任务未完成
Error.INST_INVALID_CALIBRATION_PARAM	2011	错误的标定参数
Error.INST_CALIBRATION_FAILED	2012	设备标定失败
Error.INST_CONVERT_BINFILE_FAILED	2013	转换 BIN 文件失败
Error.INST_INVALID_PARAMETER	2014	错误的参数
Error.INST_TERMINATION_CONFIG_UNSUPPORTED	2015	不支持阻抗参数配置
Error.INST_ENABLE_CHANNEL_FAILED	2016	通道使能失败
Error.INST_DISABLE_CHANNEL_FAILED	2017	通道禁能失败
Error.INST_SET_CHANNELS_ENABLED_FAILED	2018	设置通道使能状态失败

【API 定义】

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

9.3.5 设备自检

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

【API 定义】

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

9.3.6 通道配置

注意事项：

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

【阻抗类型定义】

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

【迟滞电压类型定义】

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

【边沿触发类型定义】

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

【API 定义】

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

				V
set_termination	channel：通道号 value：阻抗类型值	bool	设置通道阻抗类型值	详见前面的 TermType 定义
set_hysteresis	channel：通道号 value：迟滞电压类型值	bool	设置通道迟滞电压类型值	详见前面的 HtsType 定义
set_inter_delay	channel：通道号 value：通道延迟值	bool	设置通道间延迟值	0~1000000ps
set_dead_time	channel：通道号 value：死时间	bool	设置通道死时间	2~130000ps
set_edge_type	channel：通道号 value：边沿触发类型	bool	设置边沿触发类型	详见前面的 EdgeType 定义
set_ad_integration_point	channel：通道号 value：积分点	bool	设置 A/D 面积积分值	0~65535
enable_channel	channel：通道号	bool	设置通道使能	
disable_channel	channel：通道号	bool	设置通道禁能	
enable_all_channels	-	bool	设置所有通道使能	
disable_all_channels	-	bool	设置所有通道禁能	
get_dac	channel：通道号	float	获取通道 DAC 值	
get_hysteresis	channel：通道号	int	获取通道迟滞电压类型值	
get_inter_delay	channel：通道号	int	获取通道间延迟值	
get_dead_time	channel：通道号	int	获取通道死时间	
get_edge_type	channel：通道号	int	获取边沿触发类型	
get_ad_integration_point	channel：通道号	int	获取 A/D 面积积分值	
is_channel_enabled	channel：通道号	bool	获取通道使能状	

			态	

9.3.7 全局配置

注意事项：

- 测试模式类型主要用于仿真和测试，正常使用采用 RAW 模式；
- 切换 TDC 时间，设备会重新进行校准，耗时较长（一般在 30s 左右）；
- 系统复位操作会重置设备所有设置，耗时较长（一般在 10s 左右），请谨慎操作；
- 板卡 ID 设置主要用于多设备并联的场景，用于区分不同设备。

【TDC 时钟来源类型定义】

定义	值	说明
ClockSource.INTERNAL_CLOCK	0	内部时钟
ClockSource.EXTERNAL_CLOCK_25MHZ	1	外部时钟（25MHz）
ClockSource.EXTERNAL_CLOCK_10MHZ	2	外部时钟（10MHz）

【数据模式定义】

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

DataMode.GLOBAL_COINS_TIMESTAMP	12	全局符合（带时间戳）模式
DataMode.SINGLE_TIMESTAMP	13	单边沿时间戳模式
DataMode.GLOBAL_COINS2	14	全局符合模式 2

【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.INTERNAL_TRIGGER	6	内部触发模式

【数据带宽类型定义】

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

【A/D 面积积分缩放类型定义】

定义	值	说明
ADAreaScale.SCALE_1	0	1 倍
ADAreaScale.SCALE_1_2	1	1/2 倍
ADAreaScale.SCALE_1_4	2	1/4 倍
ADAreaScale.SCALE_1_8	3	1/8 倍
ADAreaScale.SCALE_1_16	4	1/16 倍
ADAreaScale.SCALE_1_32	5	1/32 倍

ADAreaScale.SCALE_1_64	6	1/64 倍
ADAreaScale.SCALE_1_128	7	1/128 倍

【数据采集方式定义】

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

【输出门类型定义】

定义	值	说明
OutputGateType.GATE_TYPE_ANY_COINS	0	Any coincidence flag
OutputGateType.GATE_TYPE_CH0_CH1_COINS	1	CH0 and CH1 coincidence flag
OutputGateType.GATE_TYPE_CH2_CH3_COINS	2	CH2 and CH3 coincidence flag
OutputGateType.GATE_TYPE_CH4_CH5_COINS	3	CH4 and CH5 coincidence flag
OutputGateType.GATE_TYPE_CH6_CH7_COINS	4	CH6 and CH7 coincidence flag
OutputGateType.GATE_TYPE_CH8_CH9_COINS	5	CH8 and CH9 coincidence flag
OutputGateType.GATE_TYPE_CH10_CH11_COINS	6	CH10 and CH11 coincidence flag
OutputGateType.GATE_TYPE_CH12_CH13_COINS	7	CH12 and CH13 coincidence flag
OutputGateType.GATE_TYPE_CH14_CH15_COINS	8	CH14 and CH15 coincidence flag
OutputGateType.GATE_TYPE_ANY_HIT	9	Any hit flag
OutputGateType.GATE_TYPE_CH0_HIT	10	CH0 hit flag
OutputGateType.GATE_TYPE_CH1_HIT	11	CH1 hit flag
OutputGateType.GATE_TYPE_CH2_HIT	12	CH2 hit flag

OutputGateType.GATE_TYPE_CH3_HIT	13	CH3 hit flag
OutputGateType.GATE_TYPE_SYSTEM_CLOCK_PLL_LOCKED	14	System clock PLL locked
OutputGateType.GATE_TYPE_I_GATE	15	i_gate
OutputGateType.GATE_TYPE_CH0_DELAY_HIT	16	CH0 delay hit
OutputGateType.GATE_TYPE_CH1_DELAY_HIT	17	CH1 delay hit
OutputGateType.GATE_TYPE_CH2_DELAY_HIT	18	CH2 delay hit
OutputGateType.GATE_TYPE_CH3_DELAY_HIT	19	CH3 delay hit
OutputGateType.GATE_TYPE_CH4_DELAY_HIT	20	CH4 delay hit
OutputGateType.GATE_TYPE_CH5_DELAY_HIT	21	CH5 delay hit
OutputGateType.GATE_TYPE_CH6_DELAY_HIT	22	CH6 delay hit
OutputGateType.GATE_TYPE_CH7_DELAY_HIT	23	CH7 delay hit
OutputGateType.GATE_TYPE_CH8_DELAY_HIT	24	CH8 delay hit
OutputGateType.GATE_TYPE_CH9_DELAY_HIT	25	CH9 delay hit
OutputGateType.GATE_TYPE_CH10_DELAY_HIT	26	CH10 delay hit
OutputGateType.GATE_TYPE_CH11_DELAY_HIT	27	CH11 delay hit
OutputGateType.GATE_TYPE_CH12_DELAY_HIT	28	CH12 delay hit
OutputGateType.GATE_TYPE_CH13_DELAY_HIT	29	CH13 delay hit
OutputGateType.GATE_TYPE_CH14_DELAY_HIT	30	CH14 delay hit
OutputGateType.GATE_TYPE_CH15_DELAY_HIT	31	CH15 delay hit
OutputGateType.GATE_TYPE_NCOINS_S0	32	NCoins S0 flag

【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	scale_type: 缩放类型	bool	设置 AD 区域缩放	详见 ADAreaScale 定义
set_acq_type	acq_type: 数据采集方式	bool	设置数据采集方式	详见 ACQType 定义
set_output_gate_type	gate_type: 输出门类型	bool	设置输出门类型	详见 OutGateType 定义
set_gate_delay	gate_delay: 门延迟值	bool	设置门延迟值	
set_single_countrate_window	time_ms:窗口时间值	bool	设置实时计数窗口时间值	范围：10ms-10000ms
set_coins_countrate_window	time_ms:窗口时间值	bool	设置符合计数窗口时间值	范围：10ms-10000ms
tdc_resetn	-	bool	TDC 复位	
system_resetn	-	bool	系统复位	
get_tdc_clock_source	-	int	获取 TDC 时钟源	
get_data_mode	-	int	获取数据模式	
get_test_mode	-	int	获取测试模式	
get_test_data_bandwidth	-	int	获取测试数据带宽	
get_board_id	-	int	获取板卡 ID	
get_coins_time_window	-	int	获取符合时间窗口	
get_ad_area_scale	-	int	获取 AD 区域缩放	

get_acq_type	-	int	获取数据采集方式	
get_output_gate_type	-	int	获取输出门类型	
get_gate_delay	-	int	获取门延迟值	
get_single_countrate_window	-	int	获取实时计数窗口时间值	单位：ms
get_coins_countrate_window	-	int	获取符合计数窗口时间值	单位：ms

9.3.8 状态查询

【设备状态定义】

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

【API 定义】

API 函数	参数	返回值	功能描述	注意事项
get_channel_num	-	int	获取通道数量	获取设备通道数，不包含参考通道
get_device_type	-	int	获取设备类型	0x1：Venus(TDC) 0x2: Mercury(多道分析仪) 0x3: Alea(真随机数生成

				器)
get_device_mode	-	int	获取设备型号	0x1: Venus Lite-32/Mercury-16/Alea-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0x10: Venus Lite 4 0x11: Venus Lite 2
get_product_sn	-	str	获取产品序列号	
get_device_status	-	ResultDeviceStatus	获取设备状态	包含温度和各电路电流，详见 ResultDeviceStatus 定义
get_device_temp	-	int	获取设备核心温度	
get_fan_speed	-	int	获取风扇转速	单位：RPM
get_fw_version	-	str	获取固件版本号	
get_sw_version	-	str	获取软件版本号	

9.3.9 网络配置

注意事项：

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

【API 定义】

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

9.3.10 数据采集

注意事项：

- 数据采集 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- API 返回失败，可以在错误回调中接收到具体的错误信息，或调用 get_last_error 方法查看错误代码和错误信息；
- 提供了实时获取当前数据的接口（请确保输出文件参数为空），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；

- 支持多种数据采集格式，可以根据实际需要，选择合适的格式类型；BIN 二进制格式数据最紧凑，数据采集效率最高。10G（万兆网）及以上高速数据采集场景，仅支持 BIN 二进制数据格式。

【数据采集格式定义】

定义	值	说明
ACQDataFormat.ACQ_DATAFORMAT_BIN	0	二进制格式
ACQDataFormat.ACQ_DATAFORMAT_HEX	1	十六进制格式
ACQDataFormat.ACQ_DATAFORMAT_DECODED	2	已解码格式

【API 定义】

API 函数	参数	返回值	说明
set_acq_dataformat	data_format	bool	设置数据采集格式类型
get_acq_dataformat	-	int	获取数据采集格式类型
acq_rawdata	data_mode：数据模式 duration：数据采集时长 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 True；False-等待内存中数据全部保存）	bool	开始进行数据采集（异步模式） <data_mode> 请详见前面的 DataMode 定义 <duration> 采集时间：单位-秒 <file_name> 未明确后缀名情况下： 二进制格式-默认后缀名.bin； 十六进制-默认后缀名.dat； 已解码格式-默认后缀名.csv <max_volumesize_mb> 如果分卷大小小于等于 0，文件不问卷；
fetch_rawdata	timeout_ms：超时时间（单位：ms）	bytes	实时读取 Raw 数据 返回 byte 数组

fetch_rawdata_hex	timeout_ms：超时时间（单位：ms）	list[str]	实时读取 Raw 数据 返回十六进制字符串数组 （按 8 个字节分组）
fetch_rawdata_hex_string	timeout_ms：超时时间（单位：ms）	str	实时读取 Raw 数据 返回十六进制字符串 （按 8 个字节加上 “\n” 分隔）
fetch_rawdata_list	timeout_ms：超时时间（单位：ms）	list[int]	实时读取 Raw 数据 返回 int list 主要给 matlab 调用
is_rawdata_processing	-	bool	是否在处理 RAW 数据（有一定延迟）
stop_acq	-	bool	停止数据采集
convert_to_hex	bin_file：二进制原始数据文件路径 hex_file:输出的十六进制文件路径	bool	如果 hex_file 输入为空，默认生成的十六进制文件在二进制文件同级目录下

9.3.11 数据分析

注意事项：

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

【稳定性指标类型定义】

定义	值	说明
StabilityMetricType.ADEV	0	Allan Deviation (ADEV)

StabilityMetricType.MDEV	1	Modified Allan Deviation (MDEV)
StabilityMetricType.TDEV	2	Time Deviation (TDEV)
StabilityMetricType.HDEV	3	Hadamard Deviation (HDEV)

【FLIM 分析类型定义】

定义	值	说明
FlimAnalysisType.FLIM_NONLOCATABLE_MODE	0	非定位模式
FlimAnalysisType.FLIM_LOCATABLE_MODE	1	定位模式

【FLIM 扫描类型定义】

定义	值	说明
FlimScanType.FLIM_SCAN_RASTER	0	平行扫描
FlimScanType.FLIM_SCAN_SERPENTINE	1	蛇形扫描

【API 定义】

API 函数	参数	返回值	功能描述	说明
get_single_countrate	channel:通道号	int	获取通道实时计数率	
get_coins_countrate	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_countrate	duration：数据采集时长 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_single_countrate_recording	-	bool	停止保存实时计数率	
output_coins_countrate	duration：数据采集时长 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_coins_countrate_recording	-	bool	停止保存符合计数率	
tof_analysis	duration：分析时长 ch1：通道 1 通道号 ch2：通道 2 通道号 coins_time_window：符合时间窗值（ps） enable_t3_data：是否开启 T3 数据 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	<enable_t3_data> 如果启用，会保存 T3 数据到文件，或者缓存中待读取 <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

set_cross_correlation_config	virtual_twin_ps：虚拟时间窗值（ps） bin_width_ps：BIN 宽度（ps） dcr1：通道 1 的暗计数率 dcr2：通道 2 的暗计数率	bool	设置互关联函数分析参数	
fetch_tof_processdata	-	List[int]	实时读取 TOF 分析数据	返回扁平 int 数组，格式为：<tdiff(ps), counts>
fetch_cross_correlation_data	-	List[float]	实时读取互关联分析数据	返回扁平 float 数组，格式为：<tdiff(ps), g2>
stop_tof_analysis	-	bool	停止 TOF 分析	

startstop_analysis	duration：分析时长
	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
	enable_t3_data：是否开启 T3 数据
	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> 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
				}

				<enable_t3_data> 如果启用，会保存 T3 数据到文件，或者缓存中待读取
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(ps), 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(ps), 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(ps), y_stop_tdiff(ps)>
stop_startstop_analysis	-	bool	停止开始终止通道时间差分析	
apply_ncoins_strategies	ncoins_strats：符合计数分析的策略集合	bool	下发 N 重符合分析策略到设备（硬件分析模式）	
get_ncoins_countrate	strategy_index：分析策略索引	int	返回指定策略的 N 重符合计数率（指定计数率统计时间窗口）	仅硬件分析模式
ncoins_analysis	duration：数据采集时长 ncoins_strats：符合计数分析的策略集合 enable_raw_data：是否记录 RAW 过程数据 folder_path：文件夹路径	bool	开始 N 重符合计数分析（软件分析模式）	<ncoins_strats> List[Tuple[str,List[int], int]] 示例： ncoins_strats=[

	file_name：文件名			(“ST0”,[0, 1, 2], 1000), # 策略名 ST0，通道[0, 1, 2]，虚拟时间窗 10000 (“ST1”，[3, 4], 1000), # 策略名 ST1，通道[3, 4]，虚拟时间窗 10000]
is_ncoins_analysis_processing	-	bool	是否在进行 N 重符合分析	可以通过本 API 去判断当前 N 重符合分析的状态，确保采集 duration 秒数据可以分析完成（N 重符合软件分析，依赖当前电脑硬件性能，分析时长会大于数据采集时长）
fetch_ncoins_rawdata	strategy_index：分析策略索引 timeout_ms：超时时间（单位：ms）	List[int]	实时读取指定通道分组的 N 重符合计数分析的过程符合数据	返回扁平 int 数组，格式为：<coins_index, start_ch, start_timestamp(ps),end_ch, end_timestamp(ps)>

fetch_ncoins_rawdata_list	strategy_index: 分析策略索引 timeout_ms：超时时间（单位：ms）	numpy list	实时读取指定通道分组的 N 重符合计数分析的过程符合数据	需要提前安装 numpy 模块
fetch_ncoins_countrate	strategy_index: 分析策略索引	List[int]	实时读取指定通道分组的计数率数据（指定计数率统计时间窗口）	返回扁平 int 数组，格式为：<index, count_rate>
stop_ncoins_analysis	-	bool	停止 N 重符合计数分析	
flim_loc_analysis	duration: 分析时长 coins_time_window: 符合时间窗值 frame_channel:帧通道号 line_channel:行通道号 pixel_channel:像素通道号 start_channel:开始通道号 stop_channel:终止通道号 scan_type:扫描模式 folder_path: 文件夹路径 file_name: 文件名	bool	FLIM 定位模式分析	
flim_nonloc_analysis	duration: 分析时长 coins_time_window: 符合时间窗值 startstop_chans: 开始终止分析列表	bool	FLIM 非定位模式分析	<startstop_chans> List[Tuple[int, int]] 示例：

	folder_path：文件夹路径 file_name：文件名			startstop_groups=[(0,1)-开始通道 0，终止通道 1， (2,3)-开始通道 2，终止通道 3]
stop_flim_analysis	-	bool	停止 FLIM 分析	
autocorrelation_analysis	duration：分析时长 ch：通道 min_tau：最小 tau(单位：s) max_tau：最大 tau（单位：s） bin_num：BIN 数量 enable_t2_data：是否开启 T2 数据 folder_path：文件夹路径 file_name：文件名	bool	单通道自关联函数（G2）分析	<enable_t2_data> 如果启用，会保存 T2 数据到文件，或者缓存中待读取
fetch_autocorrelation_data	-	List[float]	实时读取自关联分析数据	返回扁平 float 数组，格式为： <tdiff(s), g2>
stop_autocorrelation_analysis	-	bool	停止自关联分析	
tot_analysis	duration：分析时长 ch：通道 enable_tot_data：	bool	过阈值时间分析	<enable_tot_data> 如果启用，会保存 TOT 脉宽数据到文件，或者缓存中待读取

	是否开启 TOT 数据 folder_path：文件夹路径 file_name：文件名			
fetch_tot_processdata	-	List[int]	实时读取 Tot Process 数据	返回扁平 int 数组，格式为：<tot(ps), counts>
stop_tot_analysis	-	bool	停止 TOT 分析	
period_analysis	duration：分析时长 decades：数据量级大小 decade_points：每个数据量的点数 average_num：平均计算次数 analysis_length：分析数据长度 fnom_hz：标称频率（Hz） sampling_interval：抽样间隔 sampling_num：抽样数量 folder_path：文件夹路径 file_name：文件名	bool	频率分析	<sampling_interval> 周期抽样间隔，单位：ms <sampling_num> 周期抽样数量 如果小于等于 0，默认值为 100 数据抽样只是为了周期数据的展示，或者定量分析。
fetch_period_rawd	timeout_ms：超时	List[int]	实时读取周期	返回扁平 int 数组，

ata	时间（单位：ms）		抽样数据	格式为：<index, period(ps)>
fetch_period_processdata	-	List[int]	实时读取周期统计分析数据	返回扁平 int 数组，格式为：<period(ps), counts>
fetch_period_stability_metric	metric_type：稳定性指标类型（详见 StabilityMetricType 定义）	List[float]	实时读取稳定性指标数据	返回扁平 int 数组，格式为：<tau(s), value>
stop_period_analysis	-	bool	停止频率分析	
ad_analysis	duration：分析时长 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	
es_analysis	duration：分析时长 ch：通道 ad_integration_point: A/D 面积积	bool	能谱分析	<ch> 必须是 A/D 通道 <enable_es_data> 如果启用，会保存 ES 能量数据到文件，或者缓存中待

	分值 ad_area_scale: A/D 面积积分缩放类型 enable_es_data: 是否开启 ES 数据 folder_path：文件夹路径 file_name：文件名			读取
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：分析时长 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	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

	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]	实时读取符合能谱分析数据	返回扁平 int 数组，格式为：<tdiff(ps), counts>
fetch_conform_es_area1_processdata	-	List[int]	实时读取符合能谱 Area1 分析数据	返回扁平 int 数组，格式为：<adc, counts>
fetch_conform_es_area2_processdata	-	List[int]	实时读取符合能谱 Area2 分析数据	返回扁平 int 数组，格式为：<adc, counts>
stop_conform_es_analysis	-	bool	停止符合能谱分析	
fetch_t2_data	timeout_ms：超时时间（单位：ms）	List[int]	实时读取 T2 数据	返回扁平 int 数组，格式为：<channel, timestamp(ps)>
fetch_t2_data_list	timeout_ms：超时时间（单位：ms）	numpy list	实时读取 T2 数据	需要提前安装 numpy 模块
fetch_t3_data	timeout_ms：超时时间（单位：ms）	List[int]	实时读取 T3 数据	返回扁平 int 数组，格式为：<index, ch1,ch2,tdiff(ps)>
fetch_t3_data_list	timeout_ms：超时时间（单位：ms）	numpy list	实时读取 T3 数据	需要提前安装 numpy 模块
fetch_tot_data	timeout_ms：超时时间（单位：ms）	List[int]	实时读取 TOT 数据	返回扁平 int 数组，格式为：<channel，tot(ps)>

fetch_tot_data_list	timeout_ms：超时时间（单位：ms）	numpy list	实时读取 TOT 数据	需要提前安装 numpy 模块
fetch_es_data	timeout_ms：超时时间（单位：ms）	List[int]	实时读取 ES 数据	返回扁平 int 数组，格式为：<channel，area>
fetch_es_data_list	timeout_ms：超时时间（单位：ms）	numpy list	实时读取 ES 数据	需要提前安装 numpy 模块
fetch_flim_loc_data	timeout_ms：超时时间（单位：ms）	List[int]	实时读取 FLIM 定位数据	返回扁平 int 数组，格式为：<frame_timestamp(ps),line_timestamp(ps),pixel_index，tdiff(ps)>
fetch_flim_loc_data_list	timeout_ms：超时时间（单位：ms）	numpy list	实时读取 FLIM 定位数据	需要提前安装 numpy 模块
fetch_flim_nonloc_data	timeout_ms：超时时间（单位：ms）	List[int]	实时读取 FLIM 非定位数据	返回扁平 int 数组，格式为：<timestamp(ps)，start_chan，stop_chan，tdiff(ps)>
fetch_flim_nonloc_data_list	timeout_ms：超时时间（单位：ms）	numpy list	实时读取 FLIM 非定位数据	需要提前安装 numpy 模块
wait_data_ready	timeout_ms	bool	确保数据分析准备完成	每次数据分析前，可以先调用这个 API，确保数据采集通路准备完成（处理数据缓存残留等问题）
clear_analysis_data	-	bool	清空数据分析缓存	

9.3.12 注意事项

1) 返回值结构：

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

2) 状态码：

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

3) 线程安全：

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

4) 版本支持

支持 Python3.7.0 及以上版本

9.3.13 示例代码

典型的应用代码示例如下：

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

        # register an error callback
```

```
    tdc.on_error(lambda e, msg: print("TDC ERROR:", e, msg))

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

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

    ok = tdc.connect(devs[0].type)
    print("connected:", ok)
    if not ok:
        raise RuntimeError("Try to connect first device failed")

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

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

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

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

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

```
        # acq raw data
        tdc.set_acq_dataformat(ACQDataFormat.ACQ_DATAFORMAT_HEX)
        res = tdc.acq_rawdata(DataMode.TIMESTAMP, 10)
        print("acq_rawdata->:", res)
        raw_bytes = tdc.fetch_rawdata(500)
        time.sleep(10)
        tdc.stop_acq()
        time.sleep(2)

        # process data analysis
        res = tdc.tof_analysis(10, 0, 1, 10000)
        print("tof_analysis ->", res)
        time.sleep(10)
        data = tdc.fetch_tof_processdata()
        tdc.stop_tof_analysis()
        time.sleep(2)
        res = tdc.tof_analysis(10, 0, 1, 10000, true, 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 获取设备接口列表后，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, serial_number)]	获取可用设备接口列表	返回设备信息结构体数组。 设备信息结构体包含类型、名称和设备接口序列号等
connect	connType：设备接口类型	logical	连接设备	具体类型定义请详见 ConnType 定义。
connectBySN	deviceSN：设备接口序列号	logical	连接设备（指定设备接口序列号）	仅支持 USB3 接口类型
disconnect	-	logical	断开当前设备连接	

9.4.4 错误处理

注意事项：

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

【API 定义】

API 函数	参数	返回值	功能描述	备注

getLastError	-	(errorCode, errorMsg)	获取最近错误信息	

9.4.5 设备自检

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

【API 定义】

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

9.4.6 通道配置

注意事项：

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

【阻抗类型定义】

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

【迟滞电压类型定义】

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

【边沿触发类型定义】

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

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
setDAC	channel：通道号 value: DAC 值	logical	设置通道 DAC 电压	-1500~1500 mV
setTermination	channel：通道号 value：阻抗类型值	logical	设置通道阻抗类型值	详见前面的 TermType 定义
setHysteresis	channel：通道号 value：迟滞电压类型值	logical	设置通道迟滞电压类型值	详见前面的 HtsType 定义
setInterDelay	channel：通道号 value：通道延迟值	logical	设置通道延迟值	0~1000000 ps
setDeadTime	channel：通道号 value：死时间	logical	设置通道死时间	2~130000ps
setEdgeType	channel：通道号 value：边沿触发类型	logical	设置边沿触发类型	详见前面的 EdgeType 定义
setADIntegrationPoint	channel：通道号 value：积分点	logical	设置 A/D 面积积分值	0~65535
enableChannel	channel：通道号	logical	设置通道使能	
disableChannel	channel：通道号	logical	设置通道禁能	
enableAllChannels	-	logical	设置所有通道使能	
disableAllChannels	-	logical	设置所有通道禁能	
getDAC	channel：通道号	double	获取通道 DAC 值	
getTermination	channel：通道号	int32	获取通道阻抗类型值	

getHysteresis	channel：通道号	int32	获取通道迟滞电压类型值	
getInterDelay	channel：通道号	int32	获取通道延迟值	
getDeadTime	channel：通道号	int32	获取通道死时间	
getEdgeType	channel：通道号	int32	获取边沿触发类型	
getADIntegrationPoint	channel：通道号	int32	获取 A/D 面积积分值	
isChannelEnabled	channel：通道号	logical	获取通道使能状态	

9.4.7 全局配置

注意事项：

- 测试模式类型主要用于仿真和测试，正常使用采用 RAW 模式；
- 切换 TDC 时钟源，设备会重新进行校准，耗时较长（一般在 30s 左右）；
- 系统复位操作会重置设备所有设置，耗时较长（一般在 10s 左右），请谨慎操作；
- 板卡 ID 设置主要用于多设备并联的场景，用于区分不同设备。

【TDC 时钟来源类型定义】

定义	值	说明
ClockSource.INTERNAL_CLOCK	0	内部时钟
ClockSource.EXTERNAL_CLOCK_25MHZ	1	外部时钟（25MHz）
ClockSource.EXTERNAL_CLOCK_10MHZ	2	外部时钟（10MHz）

【数据模式定义】

定义	值	说明
DataMode.TIMESTAMP	0	时间模式
DataMode.TIME ZONE	1	时区模式
DataMode.AD	2	A/D 模式
DataMode.AREA	3	面积测量模式
DataMode.REF_COINS	4	参考符合模式

DataMode.GLOBAL_COINS	5	全局符合模式
DataMode.TOT	6	过阈时间测量模式
DataMode.AREA_COINS	7	能谱-全局符合模式
DataMode.DUAL_TIMESTAMP	8	双沿时间模式
DataMode.PERIOD	9	周期测量模式
DataMode.SINGLE_COINS	11	单符合模式
DataMode.GLOBAL_COINS_TIMESTAMP	12	全局符合（带时间戳）模式
DataMode.SINGLE_TIMESTAMP	13	单边沿时间戳模式
DataMode.GLOBAL_COINS2	14	全局符合模式 2

【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.INTERNAL_TRIGGER	6	内部触发模式

【数据带宽类型定义】

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

【A/D 面积积分缩放类型定义】

定义	值	说明

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

【数据采集方式定义】

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

【输出门类型定义】

定义	值	说明
OutputGateType.GATE_TYPE_ANY_COINS	0	Any coincidence flag
OutputGateType.GATE_TYPE_CH0_CH1_COINS	1	CH0 and CH1 coincidence flag
OutputGateType.GATE_TYPE_CH2_CH3_COINS	2	CH2 and CH3 coincidence flag
OutputGateType.GATE_TYPE_CH4_CH5_COINS	3	CH4 and CH5 coincidence flag
OutputGateType.GATE_TYPE_CH6_CH7_COINS	4	CH6 and CH7 coincidence flag
OutputGateType.GATE_TYPE_CH8_CH9_COINS	5	CH8 and CH9 coincidence flag
OutputGateType.GATE_TYPE_CH10_CH11_COINS	6	CH10 and CH11 coincidence flag
OutputGateType.GATE_TYPE_CH12_CH13_COINS	7	CH12 and CH13 coincidence flag

OutputGateType.GATE_TYPE_CH14_CH15_COINS	8	CH14 and CH15 coincidence flag
OutputGateType.GATE_TYPE_ANY_HIT	9	Any hit flag
OutputGateType.GATE_TYPE_CH0_HIT	10	CH0 hit flag
OutputGateType.GATE_TYPE_CH1_HIT	11	CH1 hit flag
OutputGateType.GATE_TYPE_CH2_HIT	12	CH2 hit flag
OutputGateType.GATE_TYPE_CH3_HIT	13	CH3 hit flag
OutputGateType.GATE_TYPE_SYSTEM_CLOCK_PLL_LOCKED	14	System clock PLL locked
OutputGateType.GATE_TYPE_I_GATE	15	i_gate
OutputGateType.GATE_TYPE_CH0_DELAY_HIT	16	CH0 delay hit
OutputGateType.GATE_TYPE_CH1_DELAY_HIT	17	CH1 delay hit
OutputGateType.GATE_TYPE_CH2_DELAY_HIT	18	CH2 delay hit
OutputGateType.GATE_TYPE_CH3_DELAY_HIT	19	CH3 delay hit
OutputGateType.GATE_TYPE_CH4_DELAY_HIT	20	CH4 delay hit
OutputGateType.GATE_TYPE_CH5_DELAY_HIT	21	CH5 delay hit
OutputGateType.GATE_TYPE_CH6_DELAY_HIT	22	CH6 delay hit
OutputGateType.GATE_TYPE_CH7_DELAY_HIT	23	CH7 delay hit
OutputGateType.GATE_TYPE_CH8_DELAY_HIT	24	CH8 delay hit
OutputGateType.GATE_TYPE_CH9_DELAY_HIT	25	CH9 delay hit
OutputGateType.GATE_TYPE_CH10_DELAY_HIT	26	CH10 delay hit
OutputGateType.GATE_TYPE_CH11_DELAY_HIT	27	CH11 delay hit
OutputGateType.GATE_TYPE_CH12_DELAY_HIT	28	CH12 delay hit
OutputGateType.GATE_TYPE_CH13_DELAY_HIT	29	CH13 delay hit
OutputGateType.GATE_TYPE_CH14_DELAY_HIT	30	CH14 delay hit
OutputGateType.GATE_TYPE_CH15_DELAY_HIT	31	CH15 delay hit
OutputGateType.GATE_TYPE_NCOINS_SO	32	NCoins S0 flag

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
setTdcClockSource	src_type:	logical	设置 TDC 时钟源	详见 ClockSource

	时钟源类型			定义
setDataMode	data_mode: 数据模式	logical	设置数据模式	详见 DataMode 定义
setTestMode	test_mode: 测试模式	logical	设置测试模式	详见 TestMode 定义
setTestDataBandwidth	bw_type: 带宽类型	logical	设置测试数据带宽	详见 TestDataBandwidth 定义
setBoardId	board_id: 板卡 ID	logical	设置新板卡 ID	0~255
setCoinsTimeWindow	value: 时间窗口	logical	设置符合时间窗口	0~16000000ps
setADAreaScale	value: 缩放类型	logical	设置 A/D 积分缩放	详见 ADAreaScale 定义
setACQType	acqType：数据采集方式	logical	设置数据采集方式	详见 ACQType 定义
setOutputGateType	gateType：输出门类型	logical	设置输出门类型	详见 OutputGateType 定义
setGateDelay	delayValue: 门延迟值	logical	设置门延迟值	
setSingleCountRateWindow	timeMs:时间窗口值	logical	设置实时计数统计窗口值	范围：10ms-10000ms
setCoinsCountRateWindow	timeMs:时间窗口值	logical	设置符合计数统计窗口值	范围：10ms-10000ms
tdcResetn	-	logical	TDC 复位	
systemResetn	-	logical	系统复位	
getTdcClockSource	-	int32	获取 TDC 时钟源	
getDataMode	-	int32	获取数据模式	
getTestMode	-	int32	获取测试模式	
getTestDataBandwidth	-	int32	获取测试数据带宽	

getBoardId	-	int32	获取板卡 ID	
getCoinsTimeWindow	-	int32	获取符合时间窗口	
getADAreaScale	-	int32	获取 A/D 积分缩放	
getACQType	-	int32	获取数据采集方式	
getOutputGateType	-	int32	获取输出门类型	
getGateDelay	-	int32	获取门延迟值	
getSingleCountRateWindow	-	int32	获取实时计数统计窗口值	单位：ms
getCoinsCountRateWindow	-	int32	获取符合计数统计窗口值	单位：ms

9.4.8 状态查询

【设备状态定义】

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

【API 定义】

API 函数	参数	返回值	功能描述	注意事项
getChannelNum	-	int32	获取通道数量	获取设备通道数，不包含参考通

				道
getDeviceType	-	int32	获取设备类型	0x1：Venus(TDC) 0x2: Mercury(多道分析仪) 0x3: Alea(真随机数生成器)
getDeviceMode	-	int32	获取设备型号	0x1: Venus Lite-32/Mercury-16/Alea-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0x10: Venus Lite 4 0x11: Venus Lite 2
getProductSN	-	char	获取产品序列号	
getDeviceStatus	-	(temperature, current_1, current_2, current_3, current_4, current_5, current_6, current_7, current_8)	获取设备状态	包含温度和各电路电流，详见前面设备状态定义
getDeviceTemp	-	Int32	返回设备核心温度	
getFanSpeed	-	Int32	返回设备风扇转速	单位：RPM
getFWVersion	-	char	返回固件版本号字符串	
getSWVersion	-	char	返回软件版本号字符串	

9.4.9 网络配置

注意事项：

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

【API 定义】

API 函数	参数	返回值	功能描述
setIPs	local_ip：本地 IP device_ip：设备 IP	logical	设置千兆网口 IP 地址
setNetPorts	local_port：本地端口 device_port：设备端口	logical	设置千兆网口端口号
setWIPs	local_ip：本地 IP device_ip：设备 IP	logical	设置万兆网口 IP 地址
setWNetPorts	local_port：本地端口 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
setWMAC	device_mac:网口 MAC 地址	logical	设置万兆网口 MAC 地址
getWMAC	-	char	获取万兆网口 MAC 地址

9.4.10 数据采集

注意事项：

- 数据采集 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑
- API 返回失败，可以调用 getLastError 方法查看错误代码和错误信息
- 提供了实时获取当前数据的接口（请确保输出文件参数为空），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
- 支持多种数据采集格式，可以根据实际需要，选择合适的格式类型；BIN 二进制格式数据最紧凑，数据采集效率最高。10G（万兆网）及以上高速数据采集场景，仅支持 BIN 二进制数据格式。

【数据采集格式定义】

定义	值	说明
ACQDataFormat.ACQ_DATAFORMAT_BIN	0	二进制格式
ACQDataFormat.ACQ_DATAFORMAT_HEX	1	十六进制格式
ACQDataFormat.ACQ_DATAFORMAT_DECODED	2	已解码格式

【API 定义】

API 函数	参数	返回值	说明
setAcqDataFormat	data_foramt：数据采集格式类型	logical	设置数据采集格式类型
getAcqDataFormat	-	int	返回数据采集格式类型
acqRawData	data_mode：数据模式 duration：数据采集时长 file_path：数据文件目录（不设置，默认是 output 目录） file_name：数据文件名	logical	开始数据采集 <data_mode> 请详见前面的 DataMode 定义 <duration>

	max_volumesize_mb: 文件分卷大小（单位: MB，默认为-1，不分卷） force_close：是否强制停止文件保存（默认为 true；false-等待内存中数据全部保存）		采集时间：单位-秒 <file_name> 未明确后缀名情况下： 二进制格式-默认后缀名.bin； 十六进制-默认后缀名.dat； 已解码格式-默认后缀名.csv
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, ...]
isRawDataProcessing	-	logical	是否在处理 RAW 数据
stopAcq	-	logical	停止数据采集
convertToHex	bin_file：二进制原始数据文件路径 hex_file:输出的十六进制文件路径	logical	如果 hex_file 输入为空，默认生成的十六进制文件在二进制文件同级目录下

9.4.11 数据分析

注意事项：

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

【API 定义】

API 函数	参数	返回值	功能描述	说明

getSingleCountRate	channel:通道号	int	获取通道实时计数率(指定计数率统计窗口)	
getCoinsCountRate	channel:通道号	int	获取通道符合计数率(指定计数率统计窗口)	必须先设置 DATA MODE 为 Global Coins (全局符合)
outputSingleCountRate	duration：数据采集时长 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”
stopSingleCountRateRecording	-	logical	停止保存实时计数率	
outputCoinsCountRate	duration：数据采集时长 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”
stopCoinsCountRateRecording	-	logical	停止保存符合计数率	
tofAnalysis	duration：分析时长 ch1：通道 1 通道号 ch2：通道 2 通道号 coins_time_window:符合时间窗值 enable_t3_data:是否开启 T3 数据 avg_times:开启平均	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 <enable_t3_data> 如果启用，会保存

	触发模式后的平均次数 folder_path：文件夹路径 file_name：文件名			T3 数据到文件，或者缓存中待读取
setCrossCorrelationConfig	virtual_twin_ps：虚拟时间窗（单位：ps） bin_width_ps：BIN 宽度（单位：ps） dcr1：通道 1 的暗计数率 dcr2：通道 2 的暗计数率	logical	设置互关联函数（G2）分析参数	
fetchTofProcessData	-	int array	实时读取 TOF 分析数据	返回扁平 int 数组，格式为：<tdiff(ps), counts>
fetchCrossCorrelationData	-	double array	实时读取互关联分析数据	返回扁平 double 数组，格式为：<tdiff(ps), g2>
stopTofAnalysis	-	logical	停止 TOF 分析	
startStopAnalysis	duration：分析时长 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 enable_t3_data:是否开启 T3 数据 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];
				<enable_t3_data> 如果启用，会保存 T3 数据到文件，或者缓存中待读取

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

			均分析数据	<tdiff(ps), counts>
fetchStartStopDualStopsRawData	start_chan：开始通道号 timeout_ms：超时时间（单位：ms）	int array	实时读取指定分组的双终止 RAW 数据	返回扁平 int 数组，格式为：<x_stop_tdiff(ps), y_stop_tdiff(ps)>
stopStartStopAnalysis	-	logical	停止开始终止通道时间差分析	
applyNCoinsStrategies	ncoins_strats：分析策略组	logical	下发 N 重符合分析策略到设备（硬件分析模式）	
getNCoinsCountRate	strategy_index：策略索引（从 1 开始）	int	获取指定策略的 N 重符合计数率（(指定计数率统计窗口)）	仅限硬件分析模式

nCoinsAnalysis	duration：数据采集时长
	ncoins_strats：分析策略组
	enable_raw_data：是否记录 RAW 过程数据
	folder_path：文件夹路径
	file_name：文件名
		logical	开始 N 重符合计数分析(软件分析模式)	<ncoins_strats>
				格式：
				struct
				array[containers.Map]
				示例 1：
				ncoins_strats[1].strategyName = “ST0”；
				ncoins_strats(1).channels = [0, 1, 2];
				ncoins_strats(1).virtualTwin = 10000;

isNCoinsAnalysisProcessing	-	logical	是否在进行 N 重符合分析	可以通过本 API 去判断当前 N 重符合分析的状态，确保采集 duration 秒数据可以分析完成（N 重符合软件分析，依赖当前电脑硬件性能，分析时长会大于数据采集时长）
fetchNCoinsRawData	strategy_index：策略索引（从 1 开始）	int array	实时读取指定分组的符合计数过程数据	返回扁平 int 数组，格式为：<coins_index, start_ch, start_timestamp(ps), end_ch, end_timestamp(ps)>
fetchNCoinsCountRate	strategy_index：策略索引（从 1 开始）	int array	实时读取指定分组的符合计数率数据((指定计数率统计窗口))	返回扁平 int 数组，格式为：<index, count_rate>
stopNCoinsAnalysis	-	logical	停止 N 重符合计数分析	
flimLocAnalysis	duration：分析时长 coins_time_window：符合时间窗值 frame_channel:帧通道号 line_channel:行通道号 pixel_channel:像素通道号 start_channel:开始通道号 stop_channel:终止通道号 bool FLIM 定位模式分析 LAB MADE EASY	logical	FLIM 定位模式分析	

	时溯科技（杭州）有限公司 154 scan_type:扫描模式 folder_path: 文件夹路径 file_name: 文件名			
flimNonLocAnalysis	duration: 分析时长 coins_time_window: 符合时间窗值 startstop_chans: 开始终止分析列表 folder_path: 文件夹路径 file_name: 文件名	logical	FLIM 非定位模式分析	<startstop_chans> Contains.Map Struct Array Cell Array 示例： startstop_chans = [struct('start_chan', 0, 'stop_chan', 1); struct('start_chan', 2, 'stop_chan', 3)];
stopFlimAnalysis	-	logical	停止 FLIM 分析	
autoCorrelationAnalysis	duration: 分析时长 ch: 通道 min_tau: 最小 tau(单位：s) max_tau: 最大 tau（单位：s） bin_num: BIN 数量 enable_t2_data: 是否开启 T2 数据 folder_path: 文件夹路径 file_name: 文件名	logical	单通道自关联函数（G2）分析	<enable_t2_data> 如果启用，会保存 T2 数据到文件，或者缓存中待读取
fetchAutoCorrelationData	-	double array	实时读取自关联分析数据	返回扁平 double 数组，格式为：<tdiff(s), g2>

stopAutoCorrelationAnalysis	-	logical	停止自关联函数分析	
totAnalysis	duration：分析时长 ch：通道 enable_tot_data：是否开启 TOT 数据 folder_path：文件夹路径 file_name：文件名	logical	过阈值时间分析	<enable_tot_data>如果启用，会保存 TOT 脉宽数据到文件，或者缓存中待读取
fetchTotProcessData	-	int array	实时读取 Tot Process 数据	返回扁平 int 数组，格式为：<tot(ps), counts>
stopTotAnalysis	-	logical	停止 TOT 分析	
periodAnalysis	duration：分析时长 decades：数据量级大小 decade_points：每个数据量的点数 average_num：平均计算次数 analysis_length：分析数据长度 fnom_hz：标称频率（Hz） sampling_interval：抽样间隔 sampling_num：抽样数量	logical	频率分析	<sampling_interval> 周期抽样间隔，单位：ms <sampling_num> 周期抽样数量 如果小于等于 0，默认值为 100 数据抽样只是为了周期数据的展示，或者定量分析。

	folder_path：文件夹路径 file_name：文件名			
fetchPeriodRawData	timeout_ms：超时时间（单位：ms）	int array	实时读取周期抽样数据	返回扁平 int 数组，格式为：<index, period(ps)>
fetchPeriodProcessData	-	int array	实时读取周期统计分析数据	返回扁平 int 数组，格式为：<period(ps), counts>
fetchPeriodStabilityMetric	metric_type：稳定性分析指标类型	double array	实时读取频率分析稳定性指标	返回扁平 double 数组，格式为：<tau(s), value>
stopPeriodAnalysis	-	logical	停止频率分析	
adAnalysis	duration：分析时长 ch：通道号 ad_integration_point：A/D 面积积分值 sampling_interval：采样间隔 sampling_num：采样数量 folder_path：文件夹路径 file_name：文件名	logical	A/D 信号分析	<ch> 必须是 A/D 通道 <sampling_interval> 采样间隔，单位：ms <sampling_num> 如果小于等于 0，全采样
fetchAdRawData	timeout_ms：超时时间（单位：ms）	int array	实时读取 A/D RAW 数据	返回扁平 int 数组，格式为：<sampling_points, adc>
stopAdAnalysis	-	logical	停止 A/D 信号分析	
esAnalysis	duration：分析时长 ch：通道 ad_integration_point：A/D 面积积分值	logical	能谱分析	<ch> 必须是 A/D 通道 <enable_es_data> 如果启用，会保存 ES 能量数据到文

	ad_area_scale: A/D 面积积分缩放类型 enable_es_data: 是否开启 ES 数据 folder_path：文件夹路径 file_name：文件名			件，或者缓存中待读取
fetchEsProcessData	-	int array	实时读取 ES 分析数据	返回扁平 int 数组，格式为：<adc, counts>
stopEsAnalysis	-	logical	停止能谱分析	
conformEsAnalysis	duration：分析时长 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 file_name：文件名	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
fetchConformEsProcessData	-	int array	实时读取符合能谱分析数据	返回扁平 int 数组，格式为：<tdiff(ps), counts>
fetchConformEsArea1ProcessData	-	int array	实时读取符合能谱 Area1 分	返回扁平 int 数组，格式为：<adc,

			析数据	counts>
fetchConformEsArea2ProcessData	-	int array	实时读取符合能谱 Area2 分析数据	返回扁平 int 数组，格式为：<adc, counts>
stopConformEsAnalysis	-	logical	停止符合能谱分析	
fetchT2Data	timeout_ms：超时时间（单位：ms）	int array	实时读取 T2 数据	返回扁平 int 数组，格式为：<channel, timestamp(ps)>
fetchT3Data	timeout_ms：超时时间（单位：ms）	int array	实时读取 T3 数据	返回扁平 int 数组，格式为：<index, ch1, ch2, tdiff(ps)>
fetchTotData	timeout_ms：超时时间（单位：ms）	int array	实时读取 TOT 脉宽数据	返回扁平 int 数组，格式为：<channel, tot(ps)>
fetchEsData	timeout_ms：超时时间（单位：ms）	int array	实时读取 ES 能量数据	返回扁平 int 数组，格式为：<channel, area>
fetchFlimLocData	timeout_ms：超时时间（单位：ms）	int array	实时读取 FLIM 定位模式数据	返回扁平 int 数组，格式为：<frame_timestamp(ps),line_timestamp(ps),pixel_index, tdiff(ps)>
fetchFlimNonLocData	timeout_ms：超时时间（单位：ms）	int array	实时读取 FLIM 非定位模式数据	返回扁平 int 数组，格式为：<timestamp(ps), start_chan, stop_chan, tdiff(ps)>
waitDataReady	timeout_ms	logical	确保数据分析准备完成	每次数据分析前，可以先调用这个 API，确保数据采集通路准备完成（处理数据缓存残留等问题）
clearAnalysisData	-	logical	清空数据分析缓存	

9.4.12 注意事项

1) **API 使用前准备：**

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

2) **状态码：**

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

3) **线程安全：**

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

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

4) **版本支持**

支持 Matlab2019b 及以上版本

9.4.13 示例代码

典型的应用代码示例如下：

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

function chronos_tdc_example()
    %% 清理环境
```

```
clear; clc; close all;

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

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

%% 2. 初始化 TDC
fprintf('\n2. Initializing TDC...\n');

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

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

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

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

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

    fprintf('TDC initialized successfully!\n');
catch ME
    fprintf('Failed to initialize TDC: %s\n', ME.message);
```

```
    fprintf('Please check:\n');
    fprintf('1. Run setup_chronos_inst to configure Python environment\n');
    fprintf('2. Python is installed and accessible\n');
    fprintf('3. chronos_inst.py is in matlab script directory\n');
    fprintf('4. Config file path and dll file path is correct\n');
    fprintf('\nTip: Run setup_chronos_inst() for automatic setup\n');
    return;
end

%% 3. 获取设备列表
fprintf('\n3. Getting device list...\n');

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

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

            %% 6. 设备自检操作示例
```

```
fprintf('\n6. Device self-check operations...\n');
result = tdc.selfCheck();
if result
fprintf('Process self-check successful\n');
else
fprintf('Process self-check falied\n');
end

%% 7. 通道配置操作示例
fprintf('\n8. Channel configuration operations...\n');
% 设置 DAC
channel = 0;
value = 100;
fprintf('Setting DAC channel %d to %.2fmV...\n', channel, value);
result = tdc.setDAC(channel, value);
if result
    fprintf('DAC set successfully: %.2fmV\n', value);
else
    fprintf('DAC set failed\n');
end
.....

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

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

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

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

%% 11. 采集控制示例
fprintf('\n12. Data acquisition control...\n');
% 设置输出 16 进制文件格式

tdc.setAcqDataFormat(ACQDataFormat.ACQ_DATAFORMAT_HEX);


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

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

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

```
            % 等待分析时间
            pause(10);

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

            %% 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 获取设备接口列表后，再进行连接；
- 操作设备前，必须先连接设备；
- 操作结束后，不再操作设备，请先断开设备连接。

【设备接口类型定义】

定义	值	说明
CI_USB3_DEV	1	Usb3.0 接口
CI_UDP_DEV	2	千兆网网口
CI_UDP_W_DEV	3	万兆网网口

【VI 定义】

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

	ChronosInst 实例 device sn：设备接口序列号 error in：错误输入	ChronosInst 实例 ret：操作状态码 error out：错误输出	定设备接口序列号，仅支持 USB3 接口）	状态码 0 -- 成功 -1 -- 失败
ciDisconnect	ChronosInst in： ChronosInst 实例 error in：错误输入	ChronosInst out： ChronosInst 实例 ret：操作状态码 error out：错误输出	断开设备连接	

9.5.5 设备自检

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

【VI 定义】

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

9.5.6 通道配置

注意事项：

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

【阻抗类型定义】

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

【迟滞电压类型定义】

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

【边沿触发类型定义】

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

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciSetDAC	ChronosInst in: ChronosInst 实例 channel：通道号 dac: DAC 设置值 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道的 DAC 值	DAC 值范围：-2000mV ~ 2000mV
ciSetTermination	ChronosInst in: ChronosInst 实例 channel：通道号 termination type: 阻抗类型值	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误	设置通道的阻抗类型值	详见前面的阻抗类型值定义（0-1）

	error in：错误输入	输出		
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 channel：通道号 A/D integration point: A/D 积分点数 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置 A/D 面积积分值	范围：0~65535
ciEnableChannel	ChronosInst in：	ChronosInst out：	设置通道使	

	ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst 实例 ret：操作状态码 error out：错误输出	能	
ciDisableChannel	ChronosInst in：ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道禁能	
ciEnableAllChannels	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置所有通道使能	
ciDisableAllChannels	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置所有通道禁能	
ciGetDAC	ChronosInst in：ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 dac: DAC 设置值 error out：错误输出	获取通道 DAC 值	
ciGetHysteresis	ChronosInst in：ChronosInst 实例	ChronosInst out：ChronosInst 实	获取通道迟滞电压类型值	

	channel：通道号 error in：错误输入	例 ret：操作状态码 hysteresis type：迟滞电压类型值 error out：错误输出		
ciGetInterDelay	ChronosInst in：ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 inter delay：时间延迟值 error out：错误输出	获取时间延迟值	
ciGetDeadTime	ChronosInst in：ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 dead time：死时间 error out：错误输出	获取通道死时间	
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	获取 A/D 面积积分值	

		ret：操作状态码 A/D integration point：A/D 积分点数 error out：错误输出		
ciIsChannelEnabled	ChronosInst in：ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 is channel enabled：通道使能状态 error out：错误输出	获取通道使能状态	

9.5.7 全局配置

注意事项：

- 测试模式类型主要用于仿真和测试，正常使用采用 RAW 模式；
- 切换 TDC 时钟源，设备会重新进行校准，耗时较长（一般在 30s 左右）；
- 系统复位操作会重置设备所有设置，耗时较长（一般在 10s 左右），请谨慎操作；
- 板卡 ID 设置主要用于多设备并联的场景，用于区分不同设备。

【TDC 时钟来源类型定义】

定义	值	说明
INTERNAL_CLOCK	0	内部时钟
EXTERNAL_CLOCK_25MHZ	1	外部时钟(25MHz)
EXTERNAL_CLOCK_10MHZ	2	外部时钟(10MHz)

【数据模式定义】

定义	值	说明

DATA_MODE_TIMESTAMP	0	时间模式
DATA_MODE_TIME ZONE	1	时区模式
DATA_MODE_AD	2	A/D 模式
DATA_MODE_AREA	3	面积测量模式
DATA_MODE_REF_COINS	4	参考符合模式
DATA_MODE_GLOBAL_COINS	5	全局符合模式
DATA_MODE_TOT	6	过阈时间测量模式
DATA_MODE_AREA_COINS	7	能谱-全局符合模式
DATA_MODE_DUAL_TIMESTAMP	8	双沿时间模式
DATA_MODE_PERIOD	9	周期测量模式
DATA_MODE_SINGLE_COINS	11	单符合模式
DATA_MODE_GLOBAL_COINS_TIMESTAMP	12	全局符合（带时间戳）模式
DATA_MODE_SINGLE_TIMESTAMP	13	单边沿时间戳模式
DATA_MODE_GLOBAL_COINS2	14	全局符合模式 2

【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_INTERNAL_TRIGGER	6	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	ChronosInst out: ChronosInst 实例 ret：操作状态码	设置 AD 面积积分缩放类型	详见 ADAreaScale 定义

	error in：错误输入	error out：错误输出		
ciSetSingleCountRateWindow	ChronosInst in：ChronosInst 实例 count rate window：时间窗口值 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置实时计数率统计窗口值	范围：10ms-10000ms
ciSetCoinsCountRateWindow	ChronosInst in：ChronosInst 实例 count rate window：时间窗口值 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置符合计数率统计窗口值	范围：10ms-10000ms
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	ChronosInst out：ChronosInst 实例	获取数据模式	

	入	ret：操作状态码 data mode：数据模式类型 error out：错误输出		
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 error out：错误输出	获取测试数据带宽	
ciGetBoardId	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 board id：板卡 ID error out：错误输出	获取板卡 ID	如果失败，会返回-1
ciGetCoinsTimeWindow	ChronosInst in：ChronosInst 实	ChronosInst out：	获取符合时间	

	例 error in：错误输入	ChronosInst 实例 ret：操作状态码 coins time window：符合时间窗值 error out：错误输出	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	
ciGetSingleCountRateWindow	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 count rate window：时间窗口值 error out：错误输出	获取实时计数率统计窗口值	单位：ms
ciGetCoinsCountRateWindow	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 count rate window：时	获取符合计数率统计窗口值	单位：ms

		间窗口值 error out：错误输出		

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:	获取设备类型	0x1：Venus(TDC) 0x2: Mercury(多道分析仪) 0x3: Alea(真随机数生成器)

		设备类型 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/Alea-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-16/Mercury-8 0x4: Venus Lite-8/Mercury-4 0x5: Venus Ultra-24 0x6: Venus Ultra-16 0x7: Venus Ultra-8 0x8: Venus Ultra Plus-12 0x9: Venus Ultra Plus-8 0xA: Venus lite4_2ps 0xB: Venus lite2_1ps
ciGetProductSN	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 product sn：设备产品序列号 error out：错误输出	获取产品序列号	
ciGetDeviceStatus	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 device status：设备状态 error out：错	获取设备状态	包含温度和各电路电流，详见前面簇定义

		误输出		
ciGetDeviceTemp	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 device temp: 设备核心温度 error out：错误输出	获取设备核心温度	
ciGetFanSpeed	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 fan speed：设备风扇转速 error out：错误输出	获取设备风扇转速	单位：RPM

9.5.9 网络配置

注意事项：

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

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciSetIPs	ChronosInst in: ChronosInst 实例 1G local IP：千兆网本地 IP	ChronosInst out: ChronosInst 实例 ret：操作状态码	设置千兆网 IP	

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

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

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

9.5.10 数据采集

注意事项：

- 数据采集 API 调用后，不会阻塞线程，API 立即返回，需要程序自己处理等待逻辑；
- 提供了实时获取当前数据的接口，可以读取数据到内存进行实时处理；
- 可以调用 stop API 去强制停止当前的数据采集任务；如果不调用，采集时间到了后，系统会自动调用 stop API，停止数据采集；
- 支持多种数据采集格式，可以根据实际需要，选择合适的格式类型；BIN 二进制格式数据最紧凑，数据采集效率最高。10G（万兆网）及以上高速数据

采

集场景，仅支持 BIN 二进制数据格式；

- 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
ciSetAcqDataFormat	ChronosInst in: ChronosInst 实例 data format:数据采集格式类型 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置数据采集格式类型	0-BIN 二进制格式 1-HEX 十六进制格式 2-DECODED 已解码格式
ciGetAcqDataFormat	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data foramt：数据采集格式类型 error out：错误输出	获取数据采集格式类型	
ciAcqRAWData	ChronosInst in: ChronosInst 实例 duration：数据采集时长 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-1，不分卷） force close：是	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），数据保存在 Server 服务器端	<data mode> 请详见前面的 DataMode 定义 <duration> 采集时间：单位-秒 <file path> 默认是 raw 目录 <file name> 未明确后缀名情况下： 二进制格式-默认后缀

	否强制停止文件保存（默认是false-等待内存中数据全部保存） error in：错误输入			名.bin；十六进制-默认后缀 名.dat；已解码格式-默认后缀 名.csv <max volume size MB> 如果分卷大小小于等于 0，文件不分卷；
ciAcqRAWDataLocal	ChronosInst in：ChronosInst 实例 duration：数据采集时长 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-1，不分卷） force close：是否强制停止文件保存（默认false-等待内存中数据全部保存） 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,数据保存在本地	同上
ciFetchRAWData	ChronosInst in：	ChronosInst out：	实时读取	

	ChronosInst 实例 timeout(ms)：超时时间（单位：毫秒） error in：错误输入	ChronosInst 实例 data：U8 字节数组 error out：错误输出	Raw 数据 返回 U8 字节数组	
ciStopAcq	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	停止数据采集	
ciConvertToHex	ChronosInst in：ChronosInst 实例 bin file path：BIN 文件路径 hex file path：输出的 HEX 文件路径 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	本地转换 BIN 文件为 HEX 文件	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
acqRAWDataToFile	ChronosInst in：ChronosInst 实例 duration：数据采集时长 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	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 Server 服务器端文件	duration delay 默认是 2 秒

	为 0） force close：是否强制停止文件保存（默认是 false） error in：错误输入			
acqRAWDataToLocalFile	ChronosInst in：ChronosInst 实例 duration：数据采集时长 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（默认是 false） error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	采集数据并保存到本地文件	duration delay 默认是 2 秒

9.5.11 数据分析

注意事项：

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

- 可以调用 stop API 去强制停止当前的分析任务；如果不调用，分析时间到了后，系统会自动调用 stop API，停止分析；
- 每个类型的分析，都提供了输出文件的接口，可以直接调用。

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciGetSingleCountRate	ChronosInst in: ChronosInst 实例 channel：指定通道号 error in：错误输入	ChronosInst out: ChronosInst 实例 single count rate: 实时计数率 error out：错误输出	获取指定通道的实时计数率值（指定实时计数率统计窗口）	
ciOutputSingleCountRate	ChronosInst in: ChronosInst 实例 duration second: 数据采集时长 folder 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	<folder path> 默认是 output 目录
ciStopSingleCountRateRecording	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	
ciGetCoinsCountRate	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（指定符合计数率统计窗口）	
ciOutputCoinsCountRate	ChronosInst in: ChronosInst 实例 duration second: 数据采集时长 folder path：数据文件目录 file name：数据文	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	<folder path> 默认是 output 目录

	件名 error in：错误输入			
ciStopCoinsCountRateRecording	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：数据分析时长 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 enable T3 data：是否启用 T3 数据 avg times：平均触发模式下的次数 folder 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 ● TOF count 分析数据文件 ● Cross correlation 分析数据（如果设置了互关联参数） ● T3 数据	<enable T3 data> 如果启用，会保存 T3 数据到文件，或者缓存中待读取 <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 <folder path> 默认是 output 目录
ciSetCrossCorrelationConfig	ChronosInst in：ChronosInst 实例 virtual twin：虚拟时间窗值 bin width：BIN 宽度 dcr1：通道 1 的暗计数率 dcr2：通道 2 的暗计数率 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置互关联分析参数	
ciFetchTofProcessD	ChronosInst in：	ChronosInst out：	实时获取 TOF	1）Process

ata	ChronosInst 实例 error in：错误输入	ChronosInst 实例 data： Int64 数值数组 error out：错误输出	Process 数据	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(ps), 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：数据分析时长 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 enable T3 data：是否启用 T3 数据 avg times：平均触发模式下的次数 folder 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 ● TOF count 分析数据文件 ● Cross correlation 分析数据（如果设置了互关联参数） ● T3 数据	<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 <folder path> 默认是 output 目录
ciApplyNCoinsStrategies	ChronosInst in: ChronosInst 实例 ncoins strats：分析	ChronosInst out: ChronosInst 实例 ret：操作状态码	下发 N 重符合计数分析策略到设备（硬件分析	

	策略组 error in：错误输入	error out：错误输出	模式）	
ciGetNCoinsCountRate	ChronosInst in：ChronosInst 实例 strategy_index：分析策略索引 error in：错误输入	ChronosInst out：ChronosInst 实例 ncoins count rate：指定策略的 N 重符合计数值 error out：错误输出	获取指定策略的 N 重符合计数值（指定符合计数率统计窗口）	仅在硬件分析模式下
ciNCoinsAnalysis	ChronosInst in：ChronosInst 实例 duration second：数据采集时长 ncoins strats：分析策略组 enable RAW data：是否记录 RAW 过程数据 folder 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	N 重符合计数分析（软件分析模式） ● 各分析策略的计数率数据文件 ● RAW 过程符合数据（如果启用了 RAW Data）	RAW 数据文件较大，保存在 Server 服务器端
ciIsNCoinsAnalysisProcessing	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 is analysis processing：是否正在分析中 error out：错误输出	是否正在 N 重符合分析	
ciFetchNCoinsCountRate	ChronosInst in：ChronosInst 实例 strategy index:策略索引 error in：错误输入	ChronosInst out：ChronosInst 实例 data： Int64 数值数组 error out：错误输出	获取指定策略的 N 重符合计数率数据（指定符合计数率统计窗口）	
ciStopNCoinsAnalysis	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码	停止 N 重符合数据分析	

		error out：错误输出		
nCoinsAnalysisToFile	ChronosInst in：ChronosInst 实例 duration：数据采集时长 duration delay：分析延迟时间 ncoins strats：分析策略组 enable RAW data：是否记录 RAW 过程数据 folder 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	N 重符合计数分析（软件分析模式）并保存到文件 ● 各分析策略的 CPS 数据文件 ● RAW 过程符合数据（如果启用了 RAW Data）	RAW 数据文件较大，保存在 Server 服务器端
ciADAnalysis	ChronosInst in：ChronosInst 实例 duration second：数据分析时长 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 folder 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	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 <folder path> 默认是 output 目录
ciFetchADRAWData	ChronosInst in：ChronosInst 实例	ChronosInst out：ChronosInst 实例	实时获取 A/D RAW 数据	1）在超时时间内，直到获

	timeout(ms)：超时时间（单位：毫秒） error in：错误输入	data： Int64 数值数组 error out：错误输出		取数据才返回； 2）返回扁平 Int64 数组，二个数字一组，格式为：sampling_points, adc
ciStopADAnalysis	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	停止 A/D 数据分析	
adAnalysisToFile	ChronosInst in：ChronosInst 实例 duration：数据分析时长 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 folder 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	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 <folder path> 默认是 output 目录
ciTotAnalysis	ChronosInst in：ChronosInst 实例 duration second：数据分析时长 ch：指定的通道号	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	TOT 数据分析并保存数据到文件 ● TOT count 分析数据文件	<enable TOT data> 如果启用，会保存 TOT 脉宽数据到文件，

	enable TOT data: 是否启用 TOT 脉宽数据 folder path：数据文件目录 file name：数据文件名 error in：错误输入		● TOT 脉宽数据	或者缓存中待读取 <folder path> 默认是 output 目录
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(ps), 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：数据分析持续时间 duration delay：分析延迟时间 ch：指定的通道号 enable TOT data: 是否启用 TOT 脉宽数据 folder 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 ● TOT count 分析数据文件	<duration delay> 默认 2 秒 <folder path> 默认是 output 目录
ciESAnalysis	ChronosInst in:	ChronosInst out:	ES 数据分析并	<ch>

	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 A/D area scale: A/D 面积积分缩放类型 enable ES data: 是否开启 ES 数据 folder path：数据文件目录 file name：数据文件名 error in：错误输入	ChronosInst 实例 ret：操作状态码 error out：错误输出	保存数据到文件 ● Process 分析数据文件 ● ES 能量数据	必须指定 A/D 通道号 <enable ES data> 如果启用，会保存 ES 能量数据到文件，或者缓存中待读取 <folder 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：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	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 <folder path> 默认是

	point: A/D 面积积分值 A/D area scale: A/D 面积积分缩放类型 enable ES data: 是否开启 ES 数据 folder path：数据文件目录 file name：数据文件名 error in：错误输入			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 energy window max2：能量窗口最大值 2 coins time	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 ● Count 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 <folder path> 默认是 output 目录 <energy window min1> <energy window max1> <energy window min2> <energy window max2> 能量窗的有效值范围：0-16384 <avg times> 大于 0，启用平均触发模式；小于等于 0，默认是单

	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：数据文件目录 file name：数据文件名 error in：错误输入			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(ps), counts
ciFetchConformESArea1ProcessData	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 Conform ES Area1 Process 数据（低通道号）	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：adc, counts
ciFetchConformESArea2ProcessData	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 Conform ES Area2 Process 数据（高通道号）	1）Process 数据是分析处理后的统计数据； 2）返回扁平 Int64 数组，二个数字一组，格式为：adc, counts
ciStopConformESA	ChronosInst in:	ChronosInst out:	停止 Conform	

nalysis	ChronosInst 实例 error in：错误输入	ChronosInst 实例 ret：操作状态码 error out：错误输出	ES 数据分析	
conformESAnalysis ToFile	ChronosInst in：ChronosInst 实例 duration：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：数据文件目录	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 <folder 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

	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 folder 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	<folder path> 默认是 output 目录
ciFetchPeriodRAWData	ChronosInst in：ChronosInst 实例 timeout(ms)：超时时间（单位：毫秒） error in：错误输入	ChronosInst out：ChronosInst 实例 data： Int64 数值数组 error out：错误输出	实时获取 Period RAW 数据	1）在超时时间内，直到获取数据才返回； 2）返回扁平 Int64 数组，二个数字一组，格式为：index, period(ps)
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(ps), 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：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） sampling number: 采样数量 folder 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	<duration delay> 默认 2 秒 <folder path> 默认是 output 目录
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 enable T3 data: 是否开启 T3 数据	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	<enable_t3_data> 如果启用，会保存 T3 数据到文件，或者缓存中待读取 <file path> 默认是 output 目录

	file path：数据文件目录 file name：数据文件名 error in：错误输入			
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(ps), y_stop_tdiff(ps)
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：tdiff(ps), 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：tdiff(ps), counts
ciStopStartStopAnalysis	ChronosInst in: ChronosInst 实例	ChronosInst out: ChronosInst 实例	停止 Start-stop 数据分析	

	error in：错误输入	ret：操作状态码 error out：错误输出		
startStopAnalysisToFile	ChronosInst in：ChronosInst 实例 duration：数据分析持续时间 duration delay：分析延迟时间 start-stop chan groups：开始终止通道分组队列 dual stops chan groups： 双终止通道分组队列 coins time window：符合时间窗值 enable T3 data：是否开启 T3 数据 folder 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	<duration delay> 默认 2 秒 <folderpath> 默认是 output 目录
ciFetchT2Data	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 T2 数据	返回扁平 int 数组，格式为：<channel, timestamp(ps)>
ciFetchT3Data	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 T3 数据	返回扁平 int 数组，格式为：<index, ch1,ch2,tdiff(ps)>

ciFetchTotData	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 TOT 脉宽数据	返回扁平 int 数组，格式为：<channel，tot(ps)>
ciFetchEsData	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 ES 能量数据	返回扁平 int 数组，格式为：<channel，area>
ciWaitDataReady	ChronosInst in: ChronosInst 实例 timeout(ms)：超时时间（ms） error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	确保数据分析准备完成	每次数据分析前，可以先调用这个 API，确保数据采集通路准备完成（处理数据缓存残留等问题）
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：

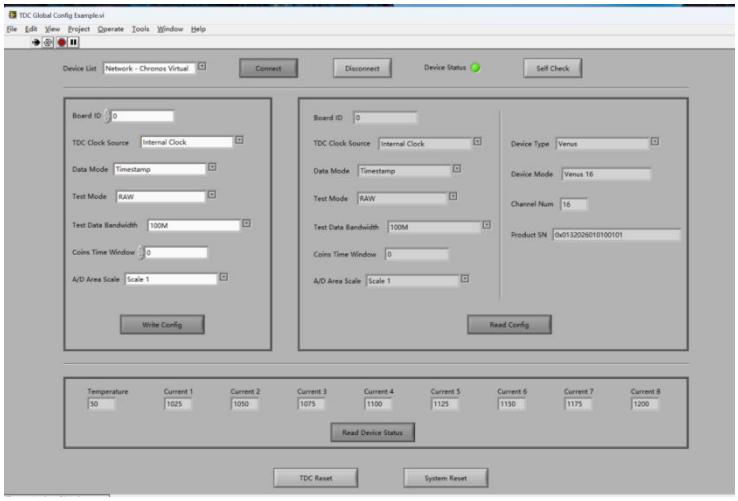
- 所有 API 调用都是线程安全的；

- 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。

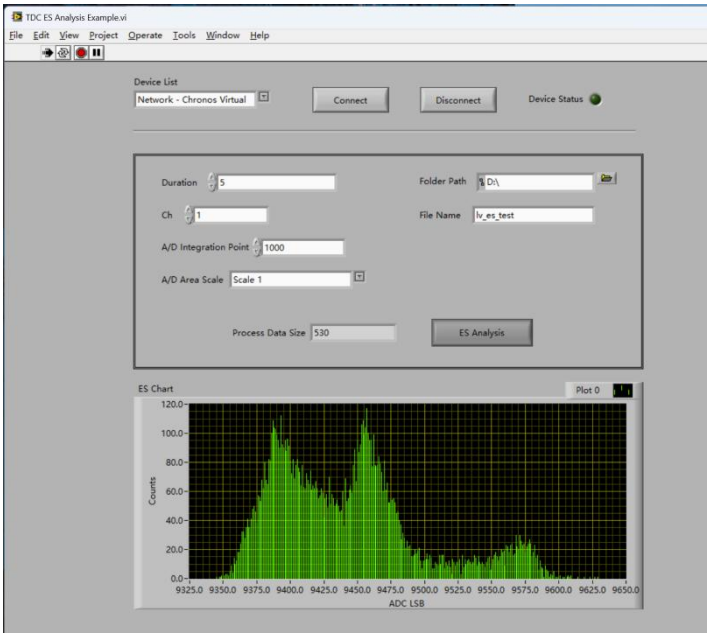
9.5.13 参考示例

请参考 Examples 目录下的示例。

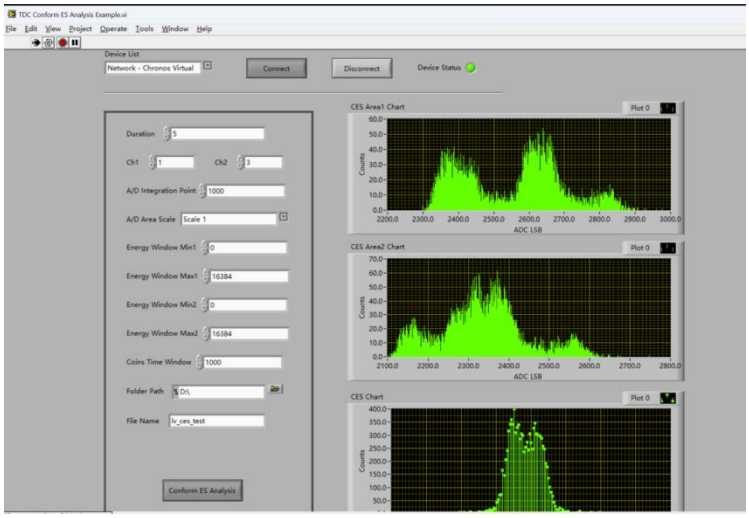
1) 全局配置 Example



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_vx.x.x”。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 9 MATLAB 原始数据分析脚本输出结果说明

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

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

9.6.2 MATLAB 分析脚本组成

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

分析程序包括四个文件夹，分别是：

（1）./dat： 保存用户使用上位机软件采集到的 Raw data；

（2）./function：保存分析程序，包括：

（2.1）dat_proc.m：主程序；

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

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

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

（2.5）channel_delay_lut.m 和 peak_seek.m：针对全局符合数据，生成各个通道延迟配置对齐值；

（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”；

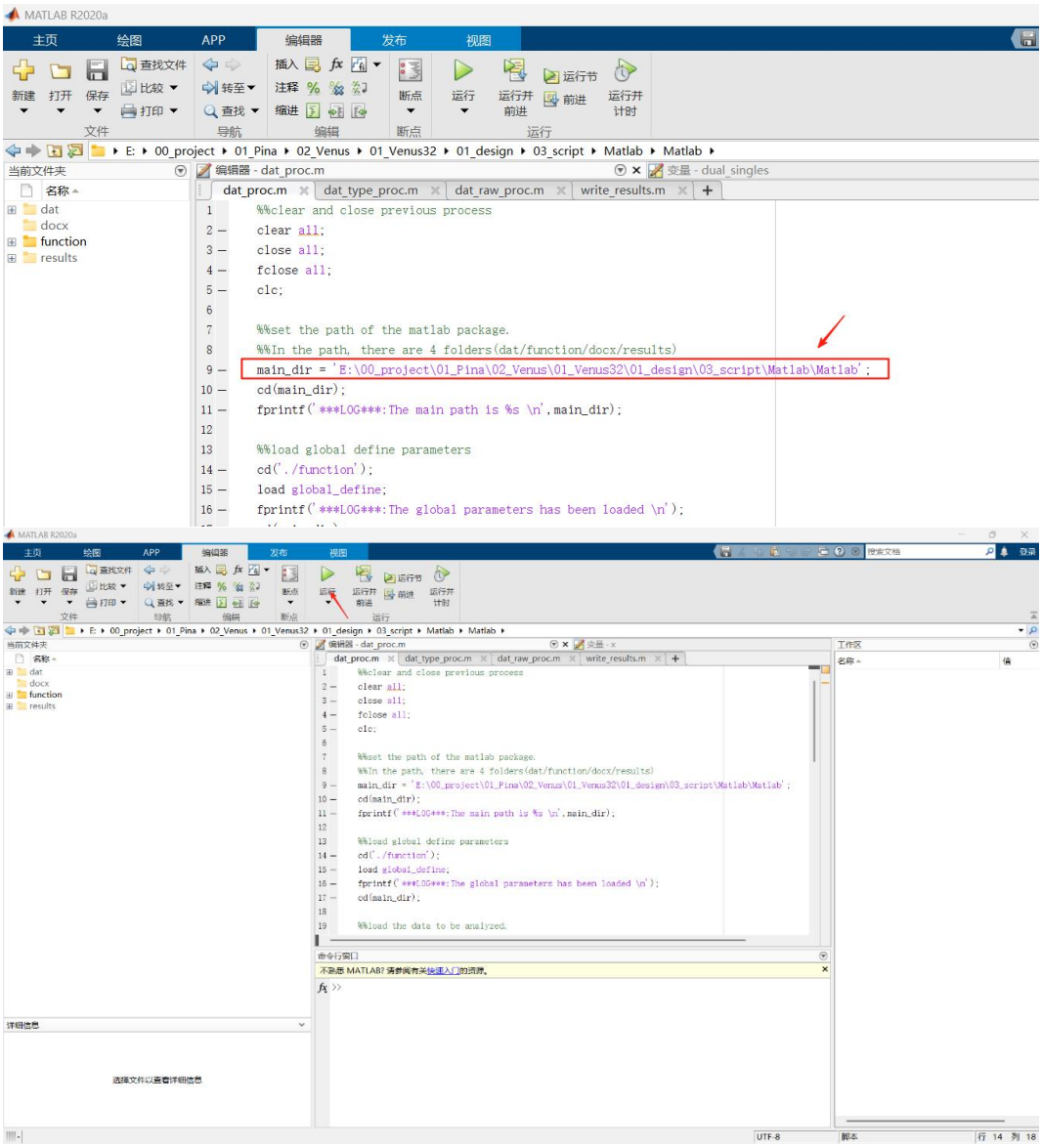


图 90 数据分析 MATLAB 脚本运行

（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”；

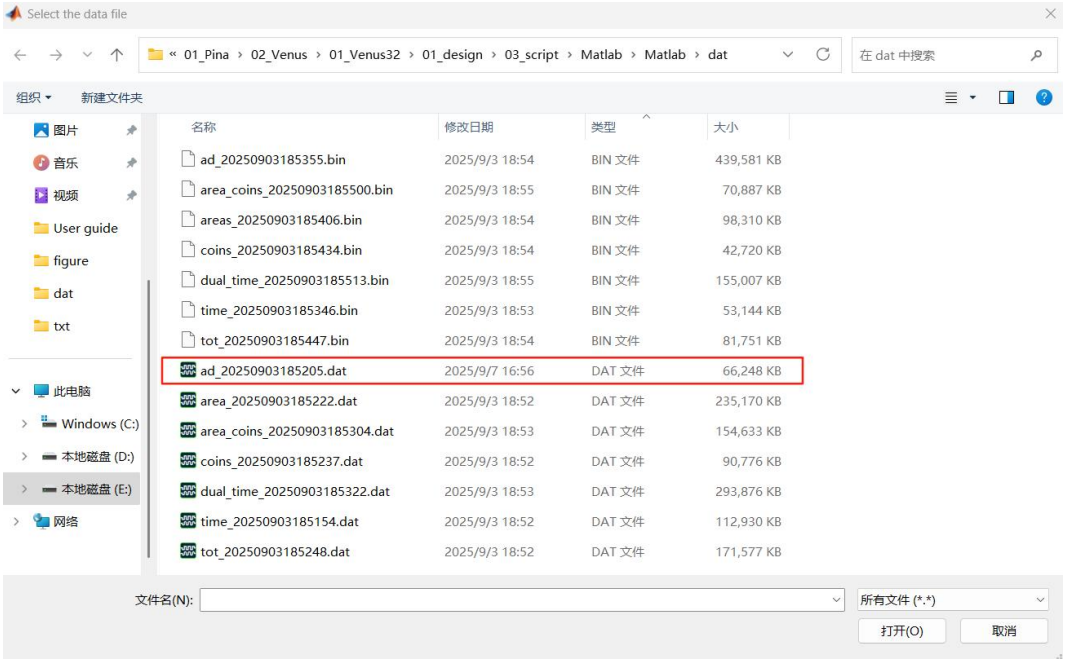


图 91 选择要分析的 adc Raw 数据文件

（3）程序会自动完成数据读取，格式转换和结果保存等；

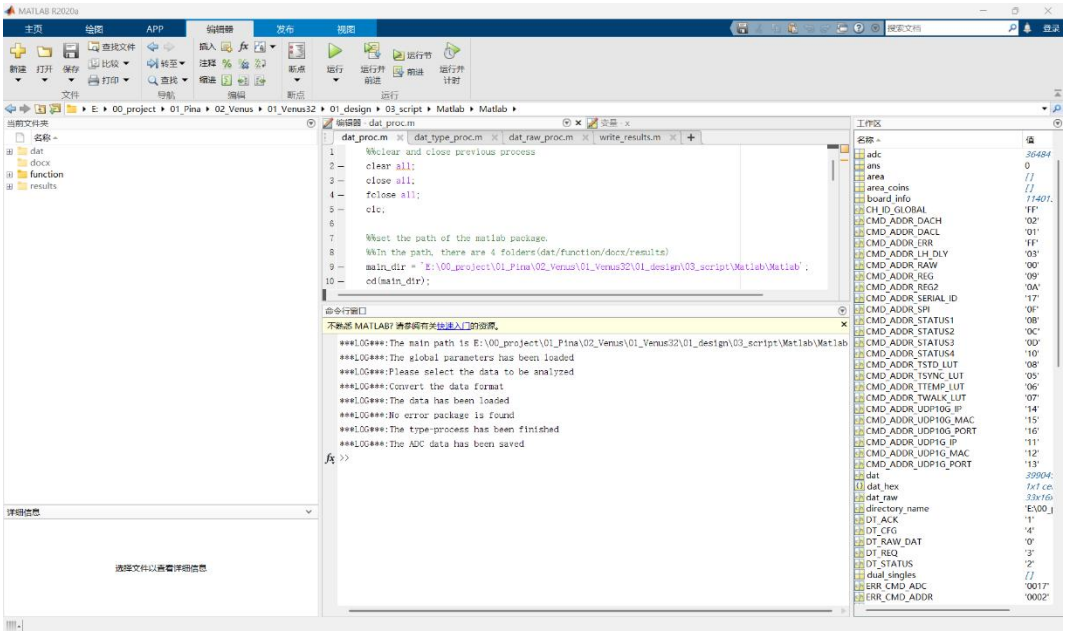


图 92 数据分析 MATLAB 脚本运行

（4）程序会自动完成数据读取，格式转换和结果保存等；

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

adc.txt

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

图 93 输出“adc.txt”数据格式

对于过阈时间数据，第一列为通道号，其他列为 TOT 计数值，单位为 ns。

tot.txt

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

图 94 输出“tot.txt”数据格式

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

area.txt

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

图 95 输出“area.txt”数据格式

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

global_coins.txt

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

图 96 输出“global_coins.txt”数据格式

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

值（T2-T1），单位为 ns。

area_coins.txt

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

图 97 输出“area_coins.txt”数据格式

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

dual_singles.txt

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

图 98 输出“dual_singles.txt”数据格式

对于周期测量数据数值，单位为 ns。

period.txt

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

图 99 输出“period.txt”数据格式

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

glbt_coins.txt

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

图 100 输出”glbt_coins.txt”数据格式

对于单沿时间戳数据，第一列为通道号，第二列为时间戳计数值，单位为 ns。

sig_singles.txt

```
Channel Timestamp (ns)
10.000000 157214941638.041626
11.000000 157214941637.661377
12.000000 157214941636.638580
5.000000 157214941641.911407
8.000000 157214941638.244415
9.000000 157214941638.256104
13.000000 157214941641.923096
14.000000 157214941641.712494
15.000000 157214941642.827881
0.000000 157214941637.498535
1.000000 157214941642.216553
2.000000 157214941641.723206
3.000000 157214941641.143097
4.000000 157214942635.430542
6.000000 157214942634.906982
7.000000 157214942635.509521
```

图 101 输出" sig_singles.txt" 数据格式

利用 Global Coins 数据，程序可以生成各个通道延迟时间值的对齐，输出参数为 Channel_delay_lut。其结构如下图所示。其中第一列为通道号，第三列为对齐各个通道需要配置的延迟值（ps）。

	1	2	3
1	0	15	8946
2	1	15	8819
3	2	15	8650
4	3	15	8488
5	4	15	7081
6	5	15	2598
7	6	15	1729
8	7	15	2677
9	8	15	1736
10	9	15	7583
11	10	15	6071
12	11	15	7777
13	12	15	872
14	13	15	7499
15	14	15	1713
16	15	15	0

图 102 Channel_delay_lut 输出

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 10 Raw 数据分析参考脚本说明

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

打开 Powershell，运行 python run_all.py 函数；

然后，程序会让用户选择待分析的数据；

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



图 103 Raw 数据分析 python 脚本分析过程

10. 典型测试电路和测试结果

10.1 测试简介

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。

由于 Venus 采用了 FPGA-TDC 技术，且每个通道电子噪声有所差异，因此本章的测试结果仅做参考典型值。实际每一台的性能官方会提供一份质量检验报告，用户具体所选购的设备具体性能可以参考质量检验报告。

10.2 双通道时间差测量（内部触发）

10.2.1 标准配置

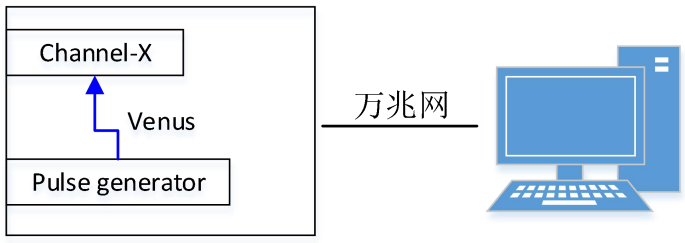


图 104 双通道时间差测量示意图（内部触发）

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。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 105 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，CH0/2）



图 106 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，CH4/5）

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 CH2 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 3.9 ps (RMS)。由此推算出的单通道定时精度为 $\sigma/\sqrt{2} \approx 2.8$ 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 8.1 ps (RMS)，对应的单通道定时精度为 $\sigma/\sqrt{2} \approx 5.7$ 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 11 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	5.5 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

在《TR-01-03-010-1 Venus 设备质量检验报告-参考样本》中的 4.1 节，用户可以得到所订购的每一台设备具体的具体测试结果，如下图所示。

4.1 CRT Performance Test(Internal Trigger)

4.1.1 CRT Performance Test(Internal Trigger, Rising Edge)

Test Result: **PASS**

Channel Group	CRT(RMS, ps)	Mean(ps)	Result
CH0/1	4.16	-323.79	PASS
CH2/3	4.47	-356.82	PASS
CH4/5	8.42	-49.63	PASS
CH6/7	7.43	193.81	PASS

图 107 Venus 设备质量检验报告-参考样本中的内部触发测试结果(没有除以√2)

10.2.2 2ps 订制版本测试

由于采用了 FPGA TDC 技术，Venus 支持将芯片资源集中用在少数通道上。在

Venus lite 4/5-2ps 特殊订制版本中，只对通道 0 - 通道 3/4 开放，而把这 4/5 个通道的时间晃动降低到 2 ps RMS。以下是典型测试结果(以 4 通道版本为例）。

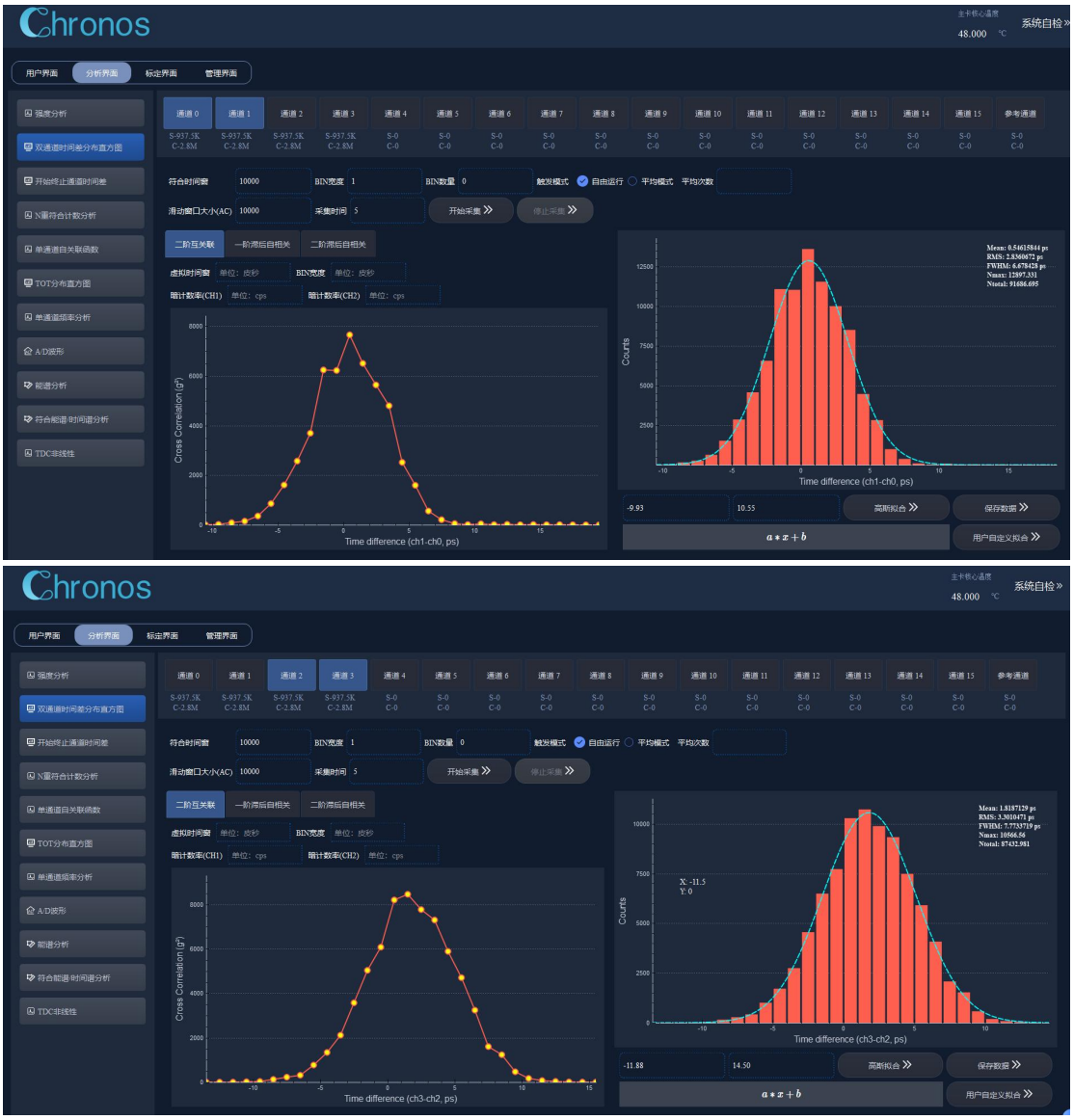


图 108 Venus Lite4-2ps 特殊版本双通道时间晃动测试（内部触发，上图：通道 0 和通道 1，下图：通道 2 和通道 3）

本次测量环境温度为 25 °C。

- 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 2.8 ps (RMS)。由此推算出的单通道定时精度为 $\sigma/\sqrt{2} \approx 1.98$ ps (RMS)。对 CH2 与 CH3 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 3.3 ps (RMS)。由此推算出的单通道定时精度为 $\sigma/\sqrt{2} \approx 2.33$ ps (RMS)。

10.3 双通道时间差测量（外部触发）

10.3.1 标准配置

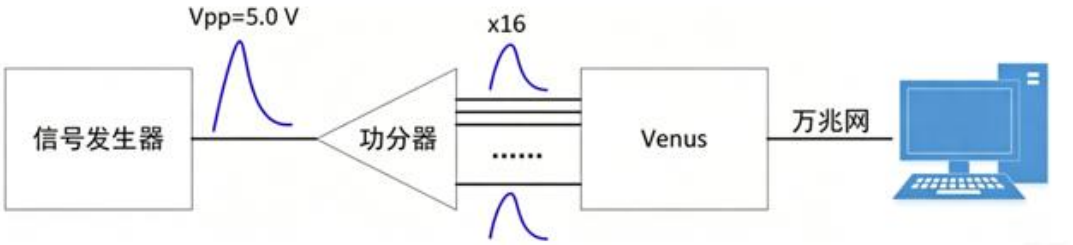


图 109 双通道时间差测量示意图

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： 0.5 MHz
  - 上升时间： 3 ns
  - 脉冲宽度： 100 ns
  - 幅度： 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 1.25 V。
- 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 110 双通道时间差外部输入信号测量结果（高分辨率通道，通道 0 和 1，上升沿，典型结果）



图 111 双通道时间差外部输入信号测量结果（普通通道，通道 8 和 9，上升沿，典型结果）

本次测量环境温度为 25 ℃。

- 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 7.6 ps (RMS)，推算的单通道定时精度为 σ/√2 ≈ 5.4 ps (RMS)；
- 普通测量通道（本次测量通道为 CH8 与 CH9）：时间差分布如下图所示，符合时间差抖动的标准差（σ）约为 9.1 ps (RMS)，推算的单通道定时精度为 σ/√2 ≈ 6.4 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。

**另外，测试结果与两路脉冲的时间延迟差相关，延迟差越大，信号源自身的晃动无法完全抵消，会导致晃动略大。必须使用高精度高速信号发生器才能基本排除信号源本身的信号晃动。由于目前测试条件受限，信号源输出信号前沿较缓，在双通道时间差较大时两路信号晃动无法完全排除。

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。且信号源本身的晃动也会影响测试结果。

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：~ 6.4 ps (RMS)
- 普通通道平均定时精度：~ 9.3 ps (RMS)

表格 12 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	5.4 ps	7.4 ps	9.0 ps	8.9 ps	6.4 ps	10.3 ps	9.6 ps	9.2 ps

在《TR-01-03-010-1 Venus 设备质量检验报告-参考样本》中的 4.2 节，用户可以得到所订购的每一台设备具体的具体测试结果，如下图所示。

4.2 CRT Performance Test(External Trigger)

4.2.1 CRT Performance Test(External Trigger, Rising Edge)

Test Result: **PASS**

Channel Group	CRT(RMS, ps)	Mean(ps)	Result
CH0/1	6.47	621.51	PASS
CH2/3	9.08	5743.04	PASS
CH4/5	11.07	198.02	PASS
CH6/7	9.92	-543.00	PASS

图 112 Venus 设备质量检验报告-参考样本中的外部触发测试结果(没有除以√2)

10.3.2 2ps 订制版本测试

由于采用了 FPGA TDC 技术，Venus 支持将芯片资源集中用在少数通道上。在 Venus 4lite-2ps 特殊订制版本中，只对通道 0-通道 3 开放，而把这 4 个通道的时间晃动降低到 2 ps RMS。以下是测试结果。





图 113 Venus Lite4-2ps 特殊版本双通道时间晃动测试（外部触发，上图：通道 0 和通道 1，下图：通道 2 和通道 3）

本次测量环境温度为 25 ℃。

- 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 6.2 ps (RMS)。由此推算出的单通道定时精度为 $\sigma/\sqrt{2} \approx 4.3$ ps (RMS)。对 CH2 与 CH3 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 8.5 ps (RMS)。由此推算出的单通道定时精度为 $\sigma/\sqrt{2} \approx 6.0$ ps (RMS)。

10.4 单通道重复频率测量

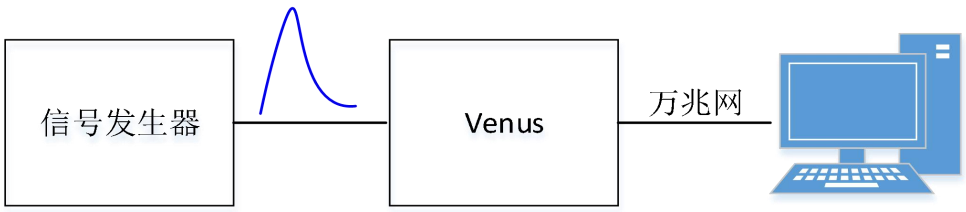


图 114 单通道周期信号重复频率测量示意图

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 10 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。

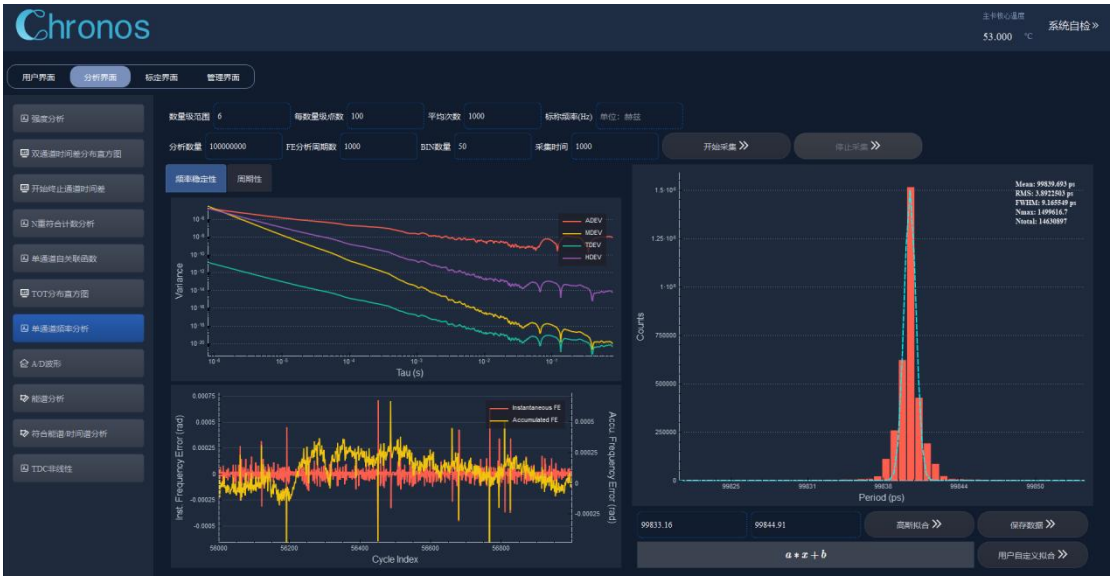


图 115 单通道重复频率测量

上位机软件计算信号周期、周期抖动（ADEV/TDEV/MDEV/HDEV）等并显示。

为了测试更高频率时钟，还测试了科迪特公司生产的 200 MHz 时钟源（KDT5351MCU-032，核心元器件为普通晶振）。



图 116 时钟方差和相位误差

除了上位机软件，通过官方提供的 API 接口，基于 Matlab/Python/LabVIEW 平台，用户可以轻松将时钟频率分析功能集成到自己的测试系统中，并得到分析结果。

10.5 单通道脉宽测量

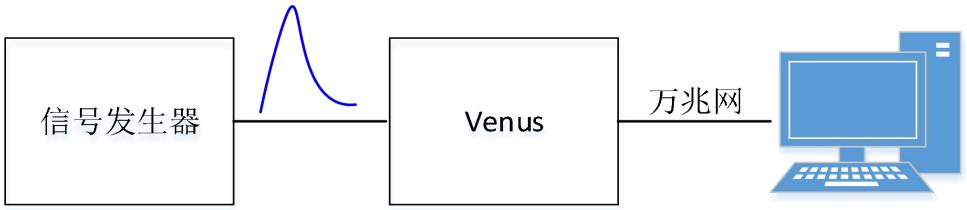


图 117 单通道信号脉宽测量示意图

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 ℃。

- 高精度测量通道（CH0）：脉宽测量平均值： 100.7 ns，脉宽晃动值：13.2 ps (RMS)；
- 普通测量通道（CH4）：脉宽测量平均值： 100.1 ns，脉宽晃动值：15.3 ps (RMS)



图 118 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 119 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：~ 11 ps (RMS)
- 普通通道平均 TOT 晃动：~13 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 13 TOT 外部输入信号典型测量结果

通道	CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7
σ	12.9 ps	10.45 ps	9.5 ps	12 ps	13.2 ps	15.8 ps	13.4 ps	12.8 ps
通道	CH8	CH9	CH10	CH11	CH12	CH13	CH14	CH15
σ	13.3 ps	14.6 ps	15.8 ps	13.7 ps	12 ps	11 ps	12.6 ps	13.8 ps

在《TR-01-03-010-1 Venus 设备质量检验报告-参考样本》中的 4.3 节，用户可以得到所订购的每一台设备具体的具体测试结果，如下图所示。

4.3 TOT Performance Test(External Trigger)

Test Result: **PASS**

Channel	TOT(RMS, ps)	Mean(ns)	Result
CH0	10.45	99.90	PASS
CH1	10.45	100.06	PASS
CH2	11.15	100.10	PASS
CH3	17.67	99.80	PASS
CH4	13.47	100.16	PASS
CH5	14.77	100.27	PASS
CH6	13.99	100.16	PASS
CH7	13.97	100.16	PASS

图 120 Venus 设备质量检验报告-参考样本中的外部触发 TOT 测试结果

10.6 模拟-数字转换实时波形测量

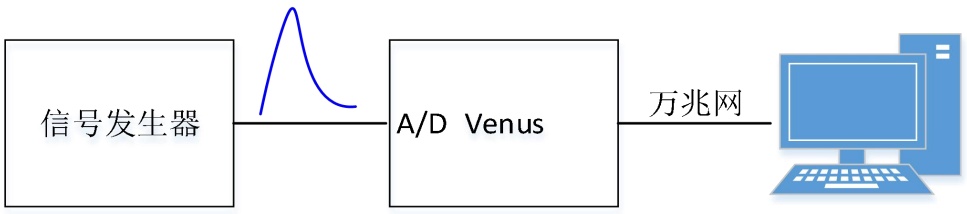


图 121 模拟-数字转换实时波形测量

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 ℃。

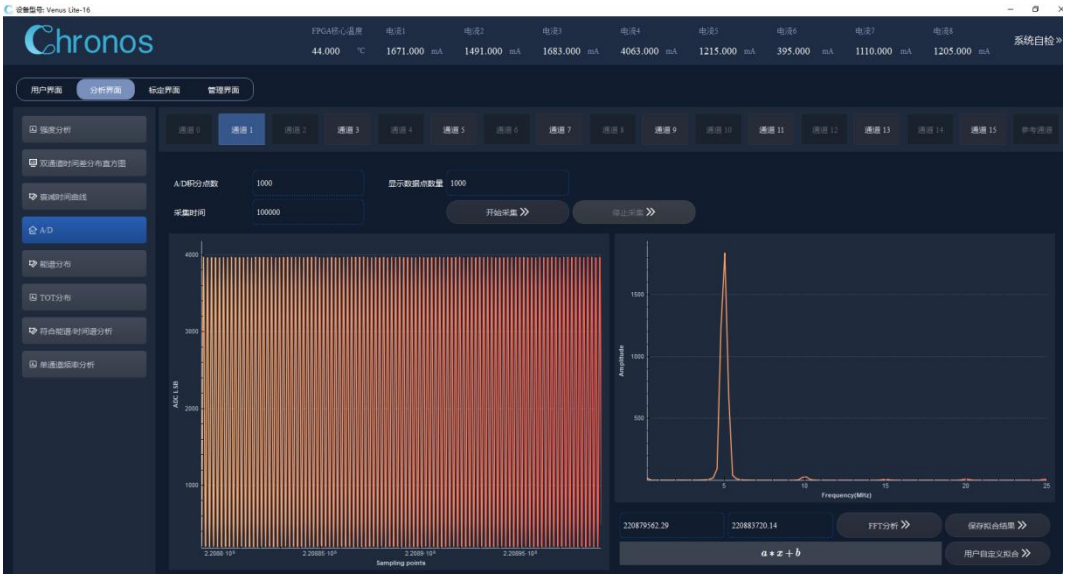


图 122 模拟-数字转换实时波形测量（输入正弦波，频率 5 MHz，幅度-2V - +2V，通道 1，典型结果）

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 ℃。

采集到的脉冲波形如下图所示。

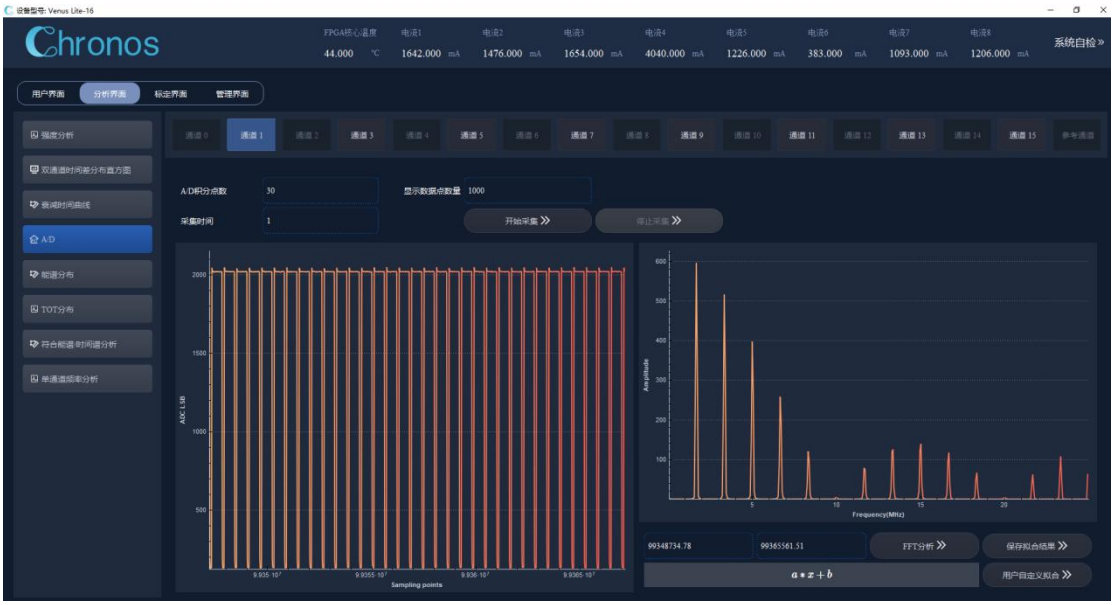


图 123 模拟-数字转换实时波形测量（输入正脉冲，频率 1 MHz，幅度+2V，通道 1，典型结果）

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 ℃。

采集到的负脉冲波形如下图所示。

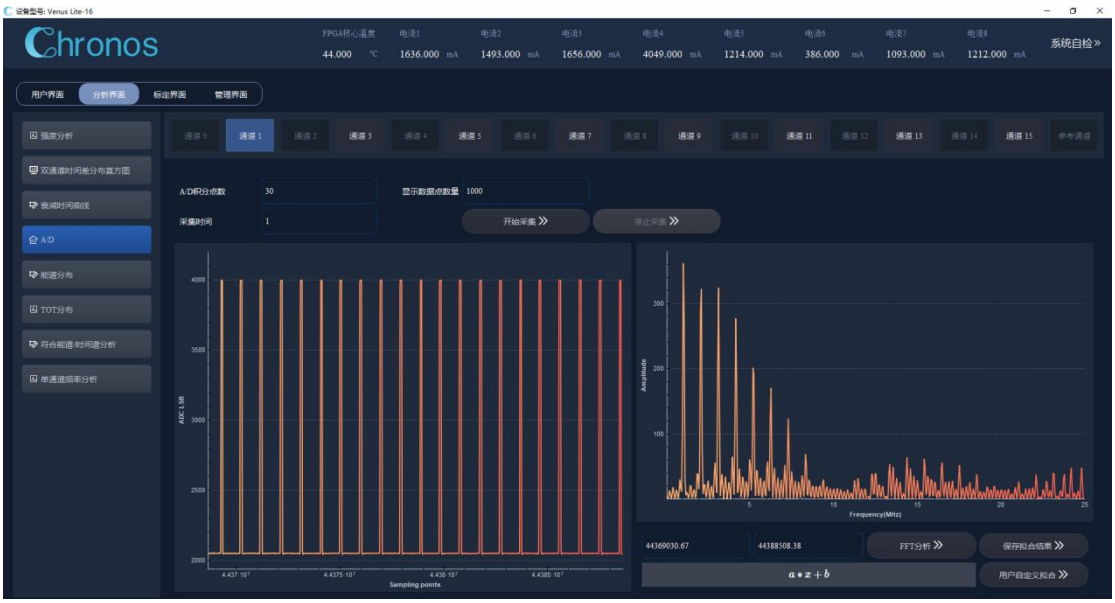


图 124 模拟-数字转换实时波形测量（输入负脉冲，频率 1 MHz，幅度-2 V，通道 1，典型结果）

10.7 能谱测量

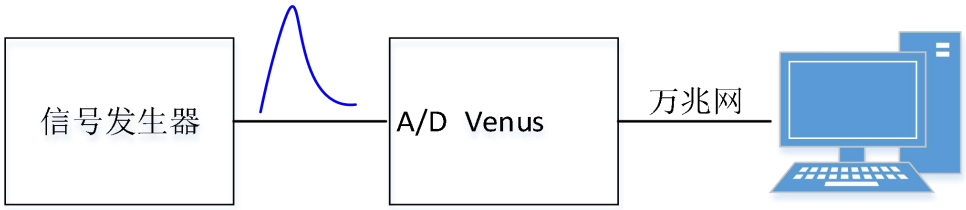


图 125 能谱测量示意图

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 ℃。

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 126 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 127 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

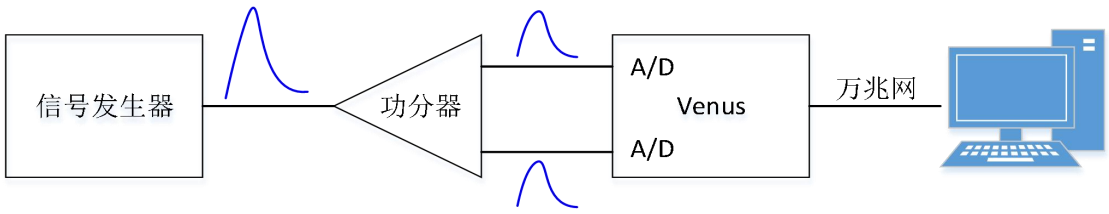


图 128 全局时间-能谱符合测量示意图

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 129 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 死时间配置测试

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 14 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

另外，还测试了向某个通道中输入 250 MHz 时钟信号，死时间默认配置（2 ns），可以收集到正常的脉冲时间戳和周期信号。

10.10 环境温度与 FPGA 核心温度关系

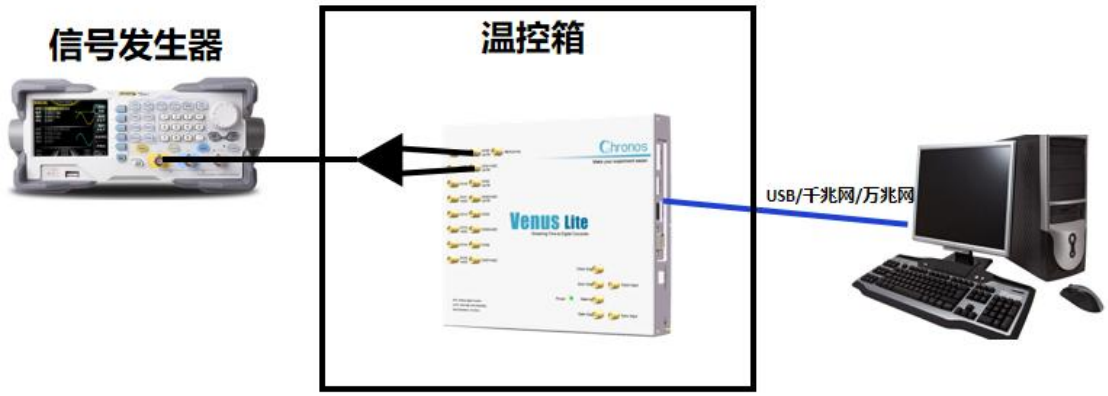


图 130 温度测试平台示意图

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。

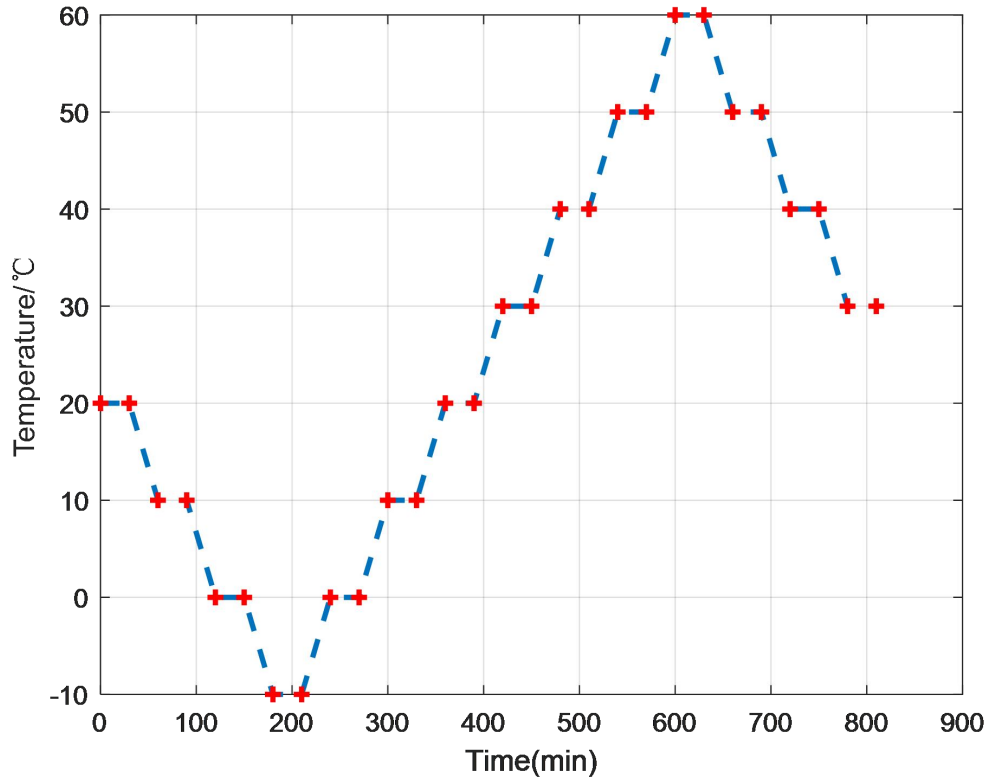


图 131 温度循环实验温度曲线设置

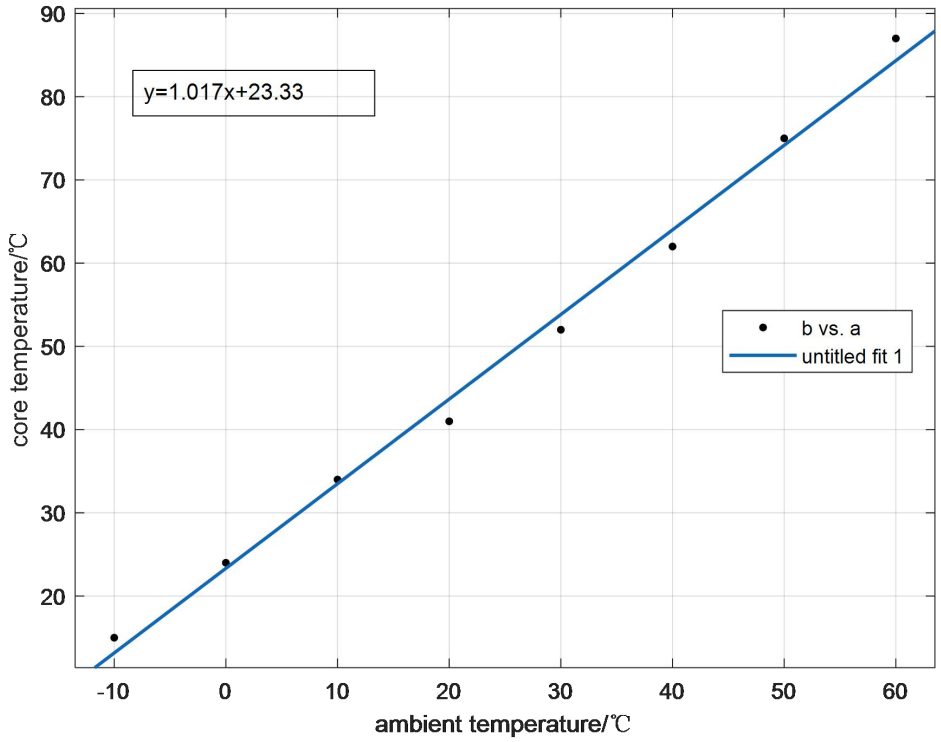


图 132 环境温度与 FPGA 内核稳定温度典型测试结果（Venus lite-T）

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。

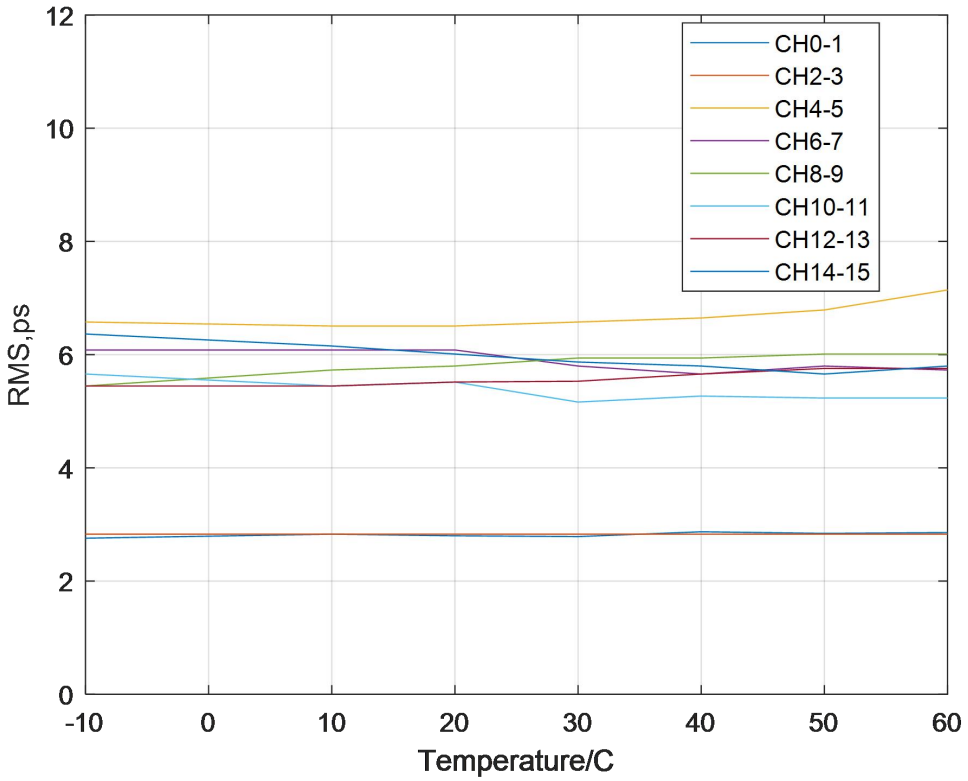


图 133 双通道时间差晃动与环境温度关系测试典型结果

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 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。

10.13 内外部时钟与时间晃动

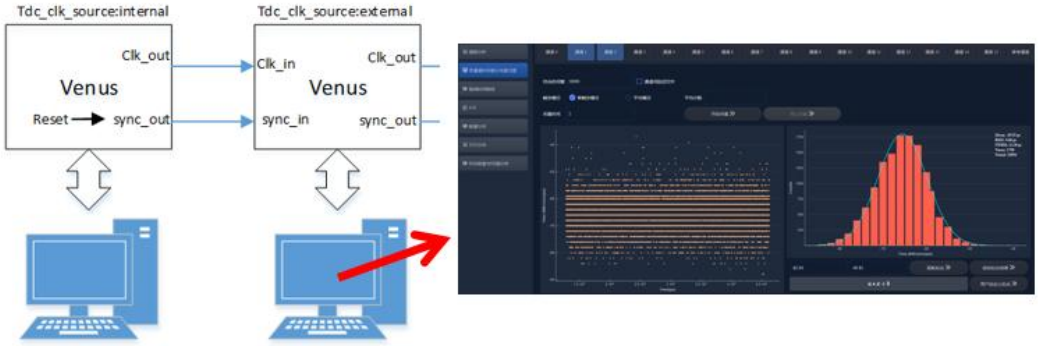


图 134 2 台 Venus 设备串联，第二台设备使用外部时钟进行时间晃动测试（内部触发）

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 135 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）

10.14 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。

10.15 能量测量线性度和精度

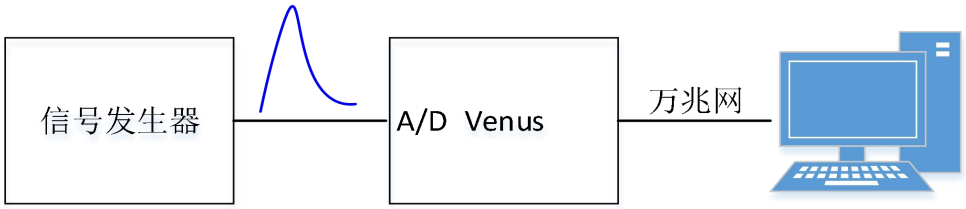


图 136 能谱测量示意图

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。

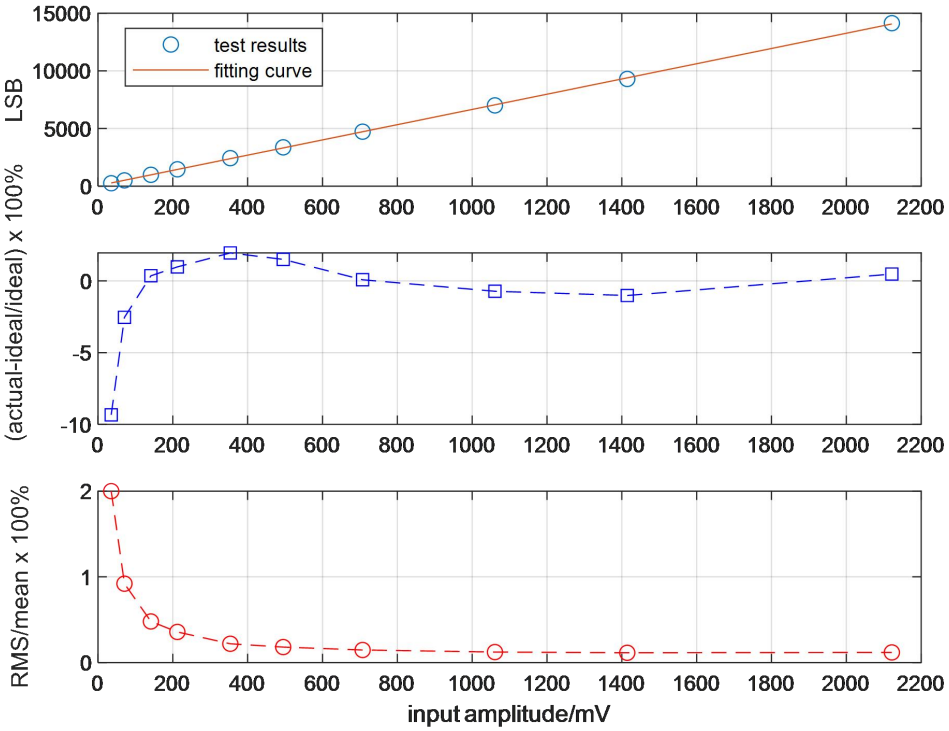


图 137 能量测量线性和分辨率测试典型结果（通道 1）

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%。

10.16 硬件在线 N 重符合测试结果

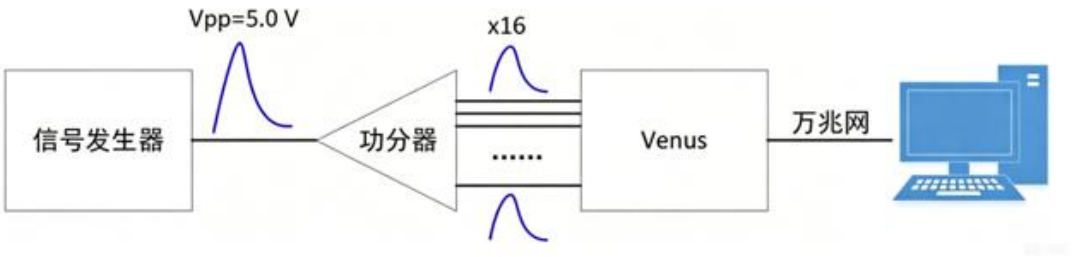


图 138 N 重符合测试平台示意图

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： 0 - 7 MHz
  - 上升时间： 3 ns
  - 脉冲宽度： 100 ns
  - 幅度： 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 1.25 V。
- 系统设置： 将这两个输入通道的时间比较阈值均设置为 500 mV，触发边沿类型为上升沿。
- 上位机选择硬件 N 重符合方式
- 上位机软件部署 40 个符合策略组，满载条件下同时运行

二、数据处理与分析

上位机软件实时统计 40 个符合策略组的计数率。

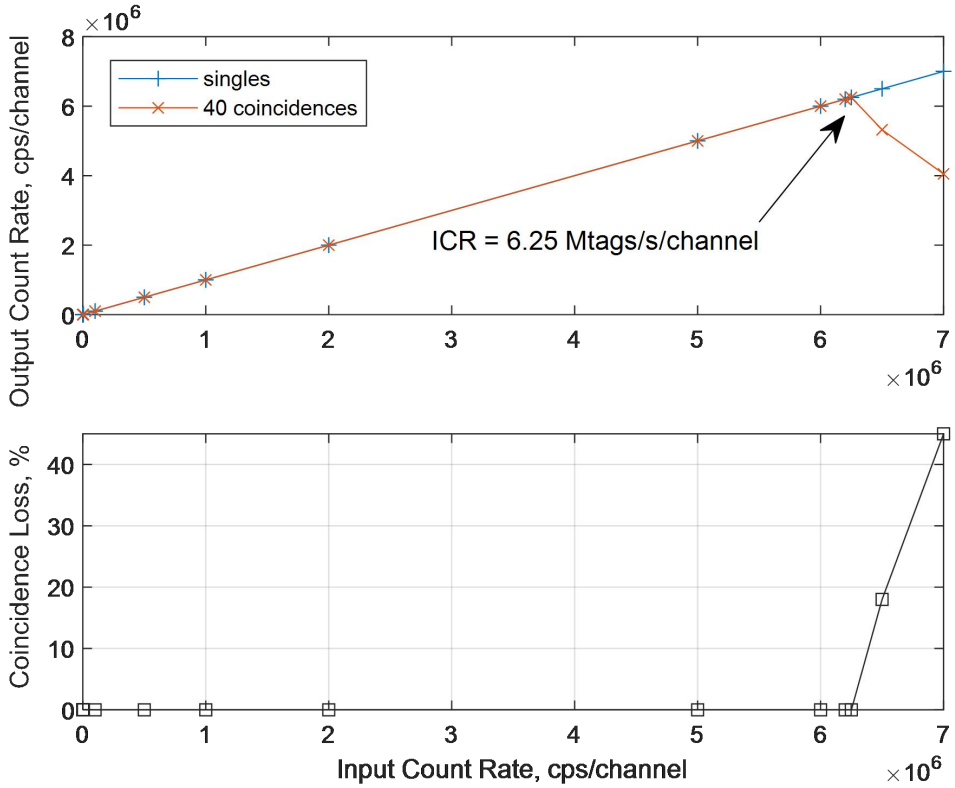


图 139 N 重符合计数率统计损失典型测试结果（硬件符合模式，40 组符合策略同时运行。上图：输入计数率与单计数率和某策略组符合计数率关系；下图：输入计数率与某策略组符合计数率损失关系）

测试结果表明，在总体计数率为 100 Mtags/s，平均各通道计数率为 100/16~6.25 Mtags/s 范围内，40 组 N 重符合计数率损伤几乎为 0。因此，硬件 N 重符合统计性能如下所示。

表格 15 硬件 N 重符合性能参数

参数	性能指标	说明
最大测量组数	40	
每个策略组最大通道数	16	Channel 0 - 15
单通道实时计数率，各通道	0 - 250 Mtags/s/channel	
N 重符合计数率，总体	100 Mtags/s (0 loss)	
N 重符合计数率，平均各通道	6.25 Mtags/s/channel (0 loss)	16 通道全部工作情况下

计数率更新周期	1 s	
各策略组时间窗设置范围	0 ~ 63 ns, LSB=0.976 ps	
软件输出	各通道实时计数率、各策略组符合计数率	
数据接口	USB/1GbE LAN/10GbE LAN	符合效率与数据接口无关
对上位机电脑配置要求	满足附录条件即可	理论上无硬性要求

10.17 Burst 测试

为了验证 TDC 在短间隔（ns 级）脉冲串条件下的时间戳记录能力：以门控 240 MHz 时钟模拟高计数率场景，输入测量通道并采集时间戳，通过相邻时间戳差值（周期）与 burst 分组统计评估系统性能。测试 Burst 脉冲样式如下图所示，每个 Burst 有 10 个脉冲信号，每个脉冲间隔约 4.166 ns，每组 Burst 重复频率为 4.166 us。



图 140 测试 Burst 脉冲样式

将测试 Burst 接入测量通道（以 CH07 为例），然后采集全部时间戳原始数据，对相邻时间戳作差得到周期序列，统计每组的命中数与组内间隔分布。一组典型的数据如下图所示，与理论预测相同，每组有 10 个时间间隔约为 4 ns 的时间差信息。Venus TDC 可以在高猝发计数输入情况下保证不丢数。

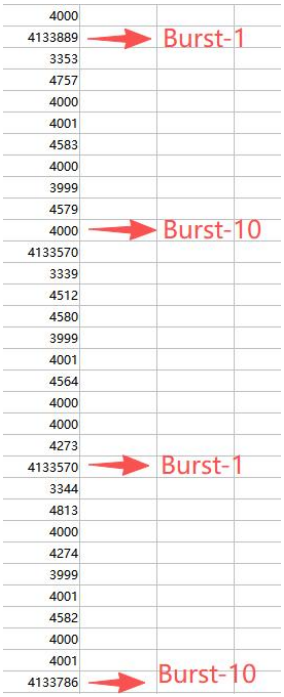


图 141 前后时间戳作差

11. 多设备时钟同步

11.1 多设备时钟同步

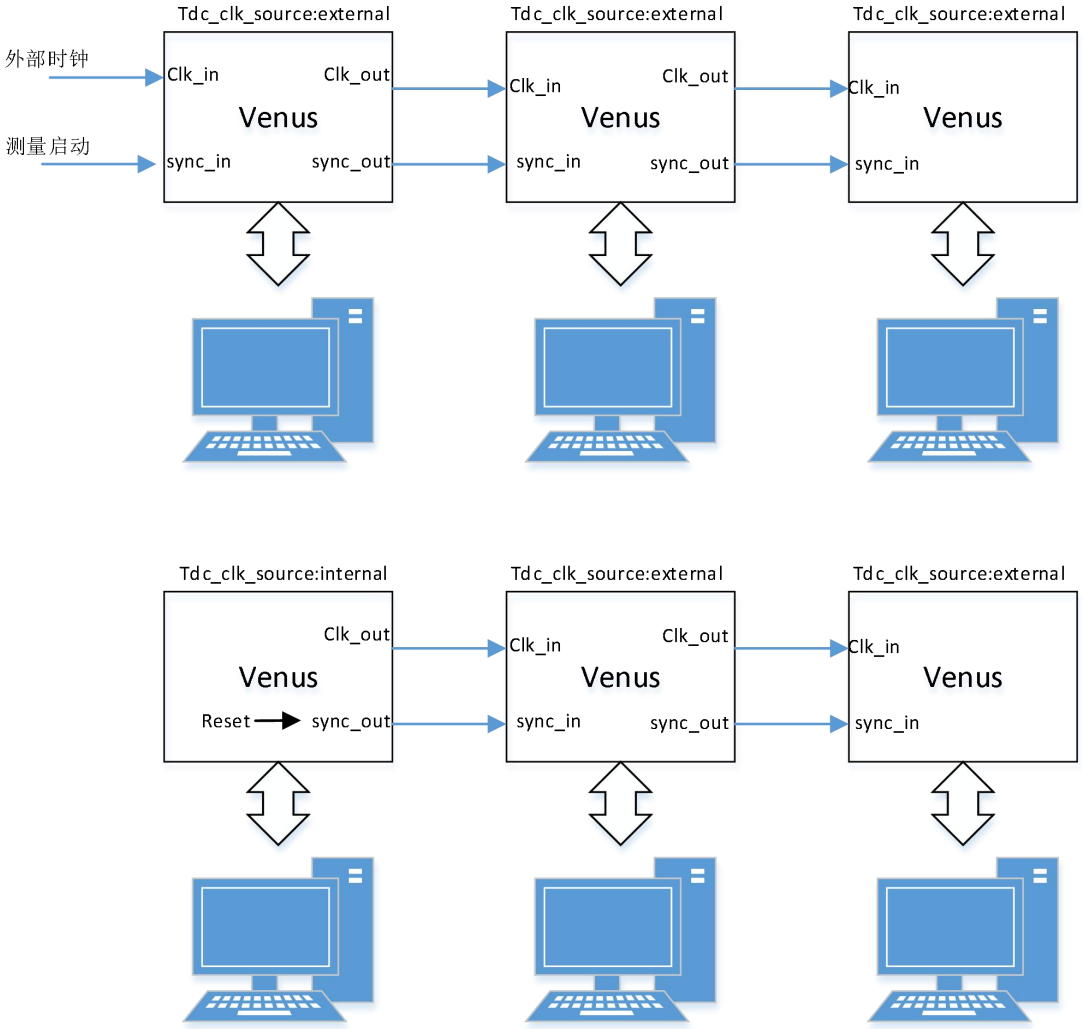


图 142 多设备时钟同步示意图（典型应用。上图：外部时钟，下图：第一个设备作为时钟源）

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，其中第一个 Venus 设备接收来自系统主时钟源的 10/25 MHz 时钟和同步信号。其他 Venus 设备必须选择 “外部时钟 25 MHz”；
- 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。设备之间的时钟频率都为 25 MHz;
- 同步触发： 当 sync_in 接收到高电平脉冲时，链上的所有 TDC 将同时复位。如果没有 Sync_in 复位信号，对第一台 Venus 设备内部 TDC 软复位也可以。第一台 Venus 设备收到 TDC 软复位命令后，会复位自身 TDC 并将 TDC 复位信号传递到下一台 Venus 设备，以此类推实现多设备 TDC 同步。

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 设备的内部本地时钟作为整个系统的主时钟源;
- 将其 25 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 “外部时钟 25 MHz” 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

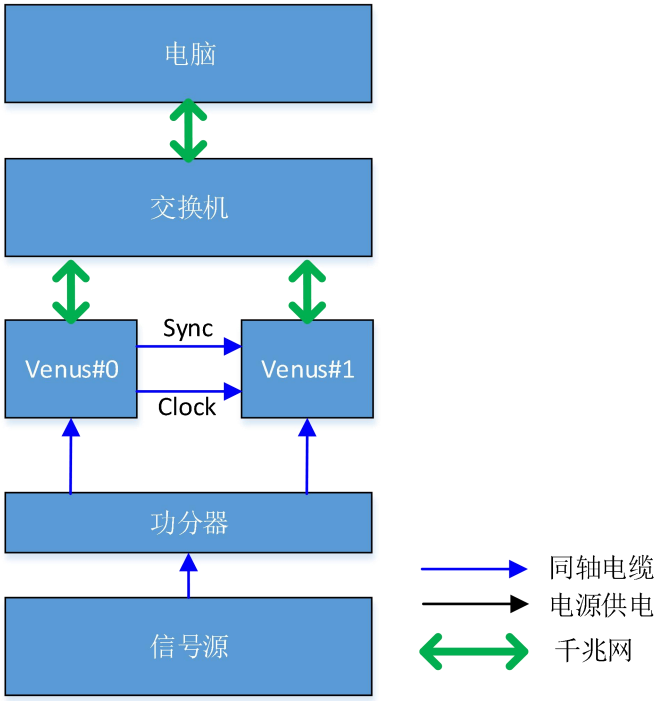


图 143 Venus 多设备组网测试示意图（以太网）

Venus 支持多设备通过以太网组网。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 16 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	0x5678	0x1234/5678
MAC 地址	0x0123456789AB	0x0123456789CD	/
板号	0x0	0x1	
TDC 时钟来源	本地时钟或外部 10/25MHz 时钟	外部时钟 25 MHz	

官方提供基于 Python 的参考脚本，名称为 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。

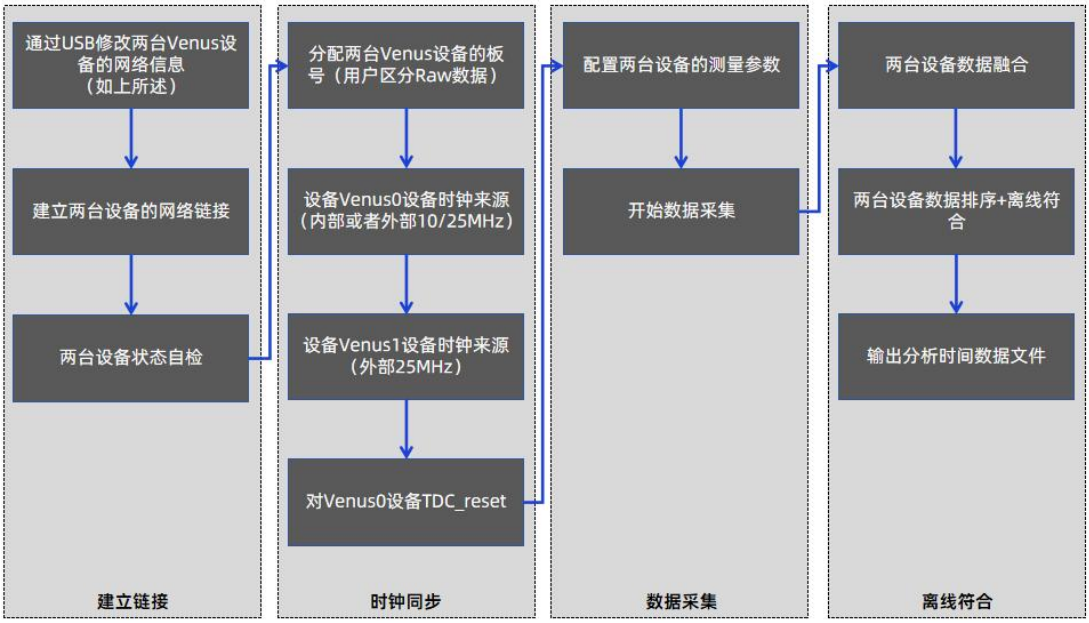


图 144 tdc_networking_example 脚本运行过程说明

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，建议通过 USB 接口修改比较稳妥。修改完网络信息后，**注意需要重启交换机操作，**否则有的交换机会有缓存导致链接错误；

（2） 在测试脚本当前目录下，新建 device0 和 device1 两个目录（分别对应板号为 0 和 1 的设备），将 apiconfig.ini 文件拷贝到两个目录中，并修改配置文件中的设备 IP 指向不同的设备。同时修改本地端口，确保两个配置文件中的本地端口不同。如下所示。

（3） 建立两台设备的网络连接；

（4） 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；

（5） 设置 Venus0 设备的 TDC 时钟来源为“本地时钟，或者外部时钟 10/25 MHz”，设置 Venus1 设备的 TDC 时钟来源为“外部时钟 25 MHz”。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 10/25 MHz”；

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

（7） 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、采集模式、采集时间等等信息；

（8） 注意，由于在切换时钟来源后，TDC 需要进行一次非线性校准，需要大概 20 s 的时间。因此，在脚本程序中，Venus1 设备在切换时钟源后，会等待一段时间，并实时检测 TDC 校准过程是否完成；

（9） 开启两台设备的数据采集，得到 Raw 数据文件，“Timestamp_raw_0_XX.bin”和 “Timestamp_raw_1_XX.bin”；

（10） 软件自动将采集到的两个设备的时间戳信息进行合并；

（11） 将合并得到的时间戳信息进行排序，得到’Timestamp_raw_sort. csv’；

（12） 用户设置分析符合时间窗值，软件根据排序的时间戳信息，以及板号和通道号信息，进行离线符合处理，得到全局符合数据文件‘Coins. csv’，包含[Board_ID-A, Channel_ID-A, Board_ID-B, Channel_ID-B , time_difference]内容，表示 A 和 B 事件的通道号和时间差值。用户可以根据符合数据进行飞行时间分析等；

（13） 符合时间窗/符合计算长度/边沿类型等用户都可以根据需要进行修改，官方提供的 API 示例代码完全开放；

（14） 官方提供的 API 中也有将.bin 文件转为.hex 格式的示例，用户可以方便转换；

（15） 用户可以使用程序中输出的任何中间数据自行进行数据分析。

分析采集到的‘Coins. csv’数据，画出两个通道（分别为板 0 的通道 0 和板 1 的通

道 1）时间差的分布图，并统计其平均值和均方根值，如下图所示。可以看到，由于同步信号延迟等原因，两个板的平均时间差较大，约为 29.5 ns，但是其时间差的晃动均方根值（/√2）约为 5.5 ps RMS，和单板测试性能接近。两个板子的延迟平均值可以通过 INTER_DELAY 寄存器进行补偿，参考 8.6.4 节。

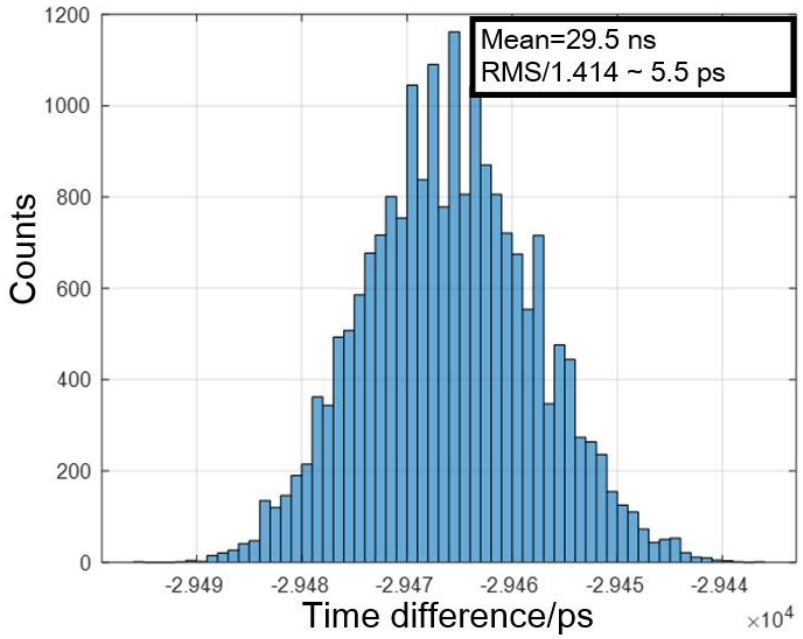


图 145 双板通道间的时间差分布直方图典型测试结果

我们在实验室将 4 台 Venus lite 设备进行了组网测试，实现 64 通道测试系统。Venus 软件支持对多达 256 个设备进行板号唯一性设置，可以搭建超过 4000 个测量通道。用户只需要按照相同的连接方式进行级联即可。Chronos 会积极配合有大量测量通道需求的用户，并提供全方位的技术支持和软件开发。需要特别注意的是，通过路由器组网测试会因为路由器数据带宽的限制导致单设备有效计数率的下降。Venus lite 组网测试还可以通过 USB 分路器、万兆网交换机等进行数据采集，其时间分辨率性能不会受到跨设备和数据接口的影响。

11.2.2 通过 USB 实现组网测试

通过 USB 接口进行多设备组网测量，基本的思路与 11.2.1 相同。使用 USB 还是网口组网，通过代码中的 USING_UDP_MODE 来选择，True 的话为以太网，False 为 USB。

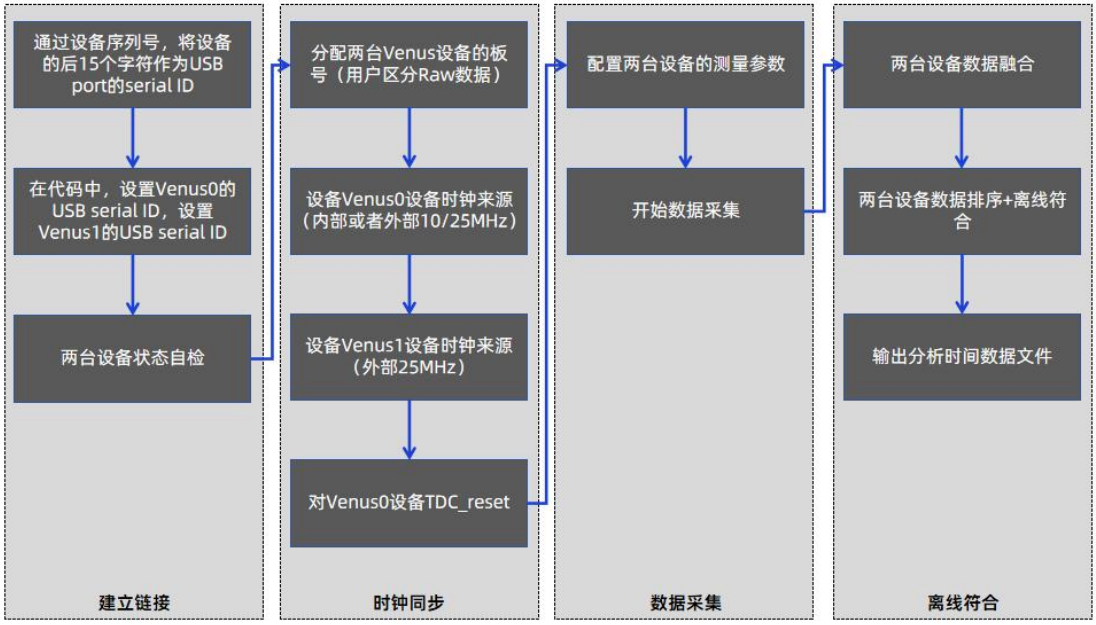


图 146 tdc_networking_example 脚本运行过程说明

注意，用户需要知道两个 USB 端口号与 Venus 设备的对应关系。请按照以下步骤设置：

（1）通过 GUI 或 API 读到两台 Venus 设备的序列号信息，假如“0x0132026090100005”和“0x0132026090100006”；

（2）设备的 USB 端口序列号为设备序列号的后 15 个字符，假如分别为“132026090100005”和“132026090100006”；

（3）对于第 0 台 Venus 设备 Venus0，假如其作为主设备，配置 MASTER_USB3_SN = "132026090100006";

（4）对于第 1 台 Venus 设备 Venus1，假如其作为从设备(其 TDC 时钟和同步信号来自 Venus0），配置 MASTER_USB3_SN = "132026090100005";

（5）正常情况下，两台设备即可建立数据连接和完成设备自检；

（6）其他操作与 11.2.1 中的（4）-（15）步骤相同。

USB 组网测试平台如下所示，测试电脑需要 2 个 USB3.0 端口。

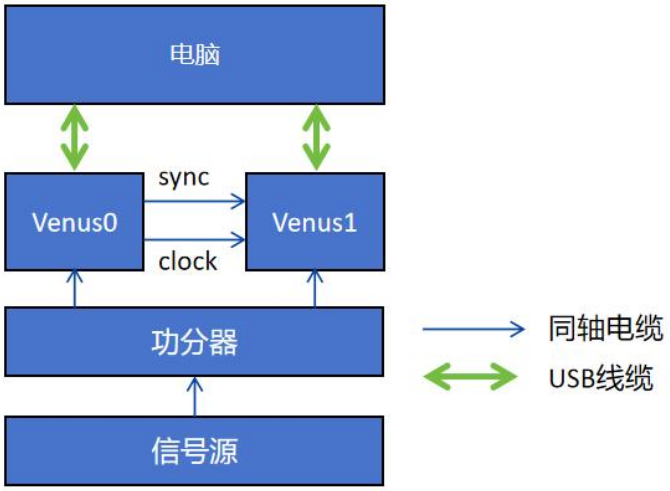


图 147 Venus 多设备组网测试示意图（USB）

11.3 并行数据获取方案

如果用户有硬件在线符合以及多板高带宽并行数据处理需求，时溯科技也会提供相应的软硬件技术方案支持服务。一种常见的多设备并行数据获取系统架构如下图所示。基于每台 Venus lite 设备拥有 QSFP 高速数据接口，每台 Venus lite 设备都可以通过最高 40 Gbps 的数据带宽与后端订制的符合板或交换板实现高速通信，然后将数据汇总后汇入到上位机电脑或用户指定的数据获取系统。具体需求需与时溯科技技术团队沟通。

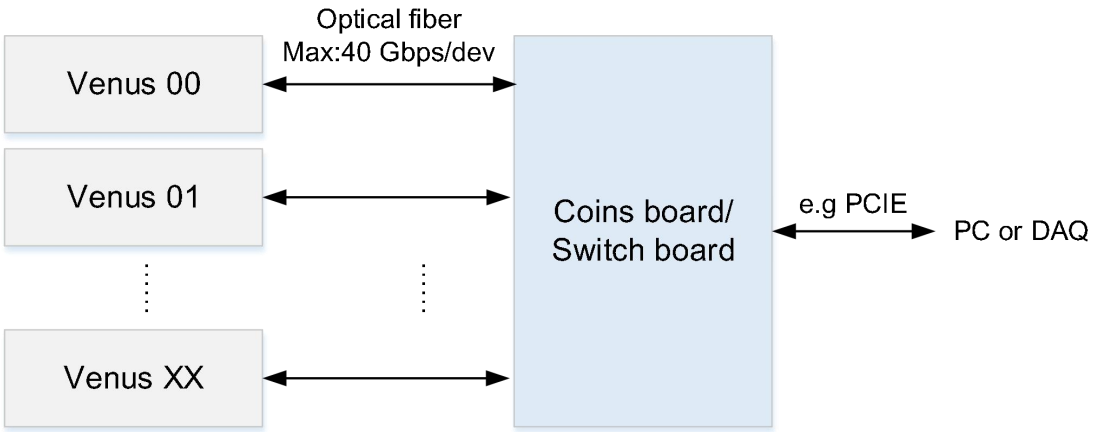


图 148 一种可能的并行数据获取系统架构

12. 测量类型原始数据结构

Venus 分为 8 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/PYTHON 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。

如果用户采集数据的时候选择了解码格式，那就不需要解码，直接用采集到的数据即可。如果用户想通过二进制或者十六进制保存数据，然后自行分析，那么可以参考本章内容进行自行解码。

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 17 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	[51:32] Frame cnt	[31:18] Product serial ID	[17:8] Firmware version	[7:0] FPGA core temperature, 0.5℃/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。

12.1 时间戳数据

表格 18 Time zone 数据格式说明

RT [63:60]：0x1	保留 [59:35]	Time zone [32:0]:time zone 时间戳 Ttz

根据 RT 判断是合理的 Time zone 数据包，提取 Time zone 时间戳信息。Time zone 时间戳根据以下公式计算。

$$t_{tz} = T_{tz} \times 2^{23} \times 0.976\ ps$$

表格 19 Event 数据戳格式说明

RT [63:60]：0x0	Event2 channel ID [59:54]	Event2 Time zone overflow flag [53]	Event2 time stamp [52:30]: LSB = 0.976 ps	Event1 channel ID [29:24]	Event1 Time zone overflow flag [23]	Event1 time stamp [22:0]: LSB = 0.976 ps

根据 RT 判断是合理的 Event 数据包，提取相关通道的 Event 时间戳信息。Event 时间戳信息与 Time zone 时间戳信息组合得到时间戳信息，根据以下公式计算。

$$t = ((T_{tz} - OVF) \times 2^{23} + T_{Events}) \times 0.976\ ps$$

其中，OVF 为 Time zone overflow flag 标志，当 OVF 为 0 时，表示此 Event 事件为本 Time zone 时区内的事件，OVF 为 1 时表示此 Event 事件发生在上一个 Time zone 时区。

特别注意：可以看到，为了提高数据传输带宽效率，每个 64 位数据包拼接了 2 个 Event 时间戳信息。为了防止 Event 时间戳跨越多个 Time zone，当一个 Time zone 内只有一个 Event 时间戳发生时，设备自动为另一个 Event 时间戳填充'1'，然后输出。例如，当在一个 Timezone 内只有一个 Event 发生，那么输出 Event 时间戳数据包如下所示。在数据分析时，把这种填充的 Event 时间戳忽略即可。

表格 20 一个时区内，只有一个时间戳发生时在另一个时间戳内全部填充 1

RT [63:60]：0x0	Event2 channel ID [59:54] :6'h3F	Event2 Time zone overflow flag [53] : 1'b1'	Event2 time stamp [52:30]: 23'h7FFFFF	Event1 channel ID [29:24]	Event1 Time zone overflow flag [23]	Event1 time stamp [22:0]: LSB = 0.976 ps

12.2 时间参考/全局符合数据

表格 21 时间参考/全局符合数据格式说明

RT [63:60] : 0x4/5	CH_ID-A 通道计数值	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 = 0.976 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。

$$t = Tdiff \times 0.976\ ps$$

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;

CH_ID-A 通道计数值为 CH_ID-A 有效计数值，即每次 CH_ID-A 通道时间数据有效时，此计数值累加。这个在 Start-stop 测试中有较大作用。比如，CH-ID-A 为 Start 通道，这个计数值可以让用户区分哪一次 Start 信号时的 Stop-Start 时间差值。

12.3 ADC 测量数据

表格 22 ADC 测量数据格式说明

RT [63:60]：0x2	Reserve [59:44]	CH_ID-A [43:36]:通道号	AD-A 值 [35:22]	CH_ID-B [21:14]:通道号	AD-B 值 [13:0]

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 23 能谱测量数据格式说明

RT [63:60]=0x3	CH_ID [59:52]	时间戳 [51:14],LSB=0.976 ps	能量值 [13:0]

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 24 脉宽测量数据格式说明

RT [63:60]=0x6	Reserve [59:32]	CH_ID [31:24]:通道号	TOT 值 [23:0]：脉宽，LSB=0.976ps

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

$$t = TOT \times 0.976\ ps$$

12.6 时间-能谱全局符合数据

表格 25 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 = 0.976 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。其中时间差的正负判断方法与 12.2 节中的一致。

$$t = Tdiff \times 0.976\ ps$$

12.7 双边沿时间戳数据

表格 26 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 T: [50:0] LSB = 0.976 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。

$$t = T \times 0.976\ ps$$

12.8 周期测量数据

表格 27 周期测量数据格式说明

RT [63:60]=0x9	[59:30]: Period1	[29:0]: Period2

根据 RT 判断是合理的周期数据包，提取周期时间。Period1 和 Period2 均为输入时钟的周期，Period1 比 Period2 时间序列靠前。

$$t_period = Period \times \frac{1}{8} \times 0.976\ ps$$

12.9 带时间戳的全局符合数据

表格 28 带时间戳的全局符合数据格式说明

RT [63:60]=0xC	[59:16]: Timestamp of Channel-A,TA, LSB = 124.8 ps	[15:8]: Channel-A ID	[7:0]: Channel-B ID

根据 RT 判断是合理的数据包，提取通道 A 的时间戳信息。

$$t = TA \times 124.8\,ps$$

12.10 单边沿时间戳数据

表格 29 单边沿时间戳测量数据格式说明

RT [63:60]=0xD	CH_ID [59:52]: 通道号	时间戳 T: [51:0] LSB = 0.976 ps

根据 RT 判断是合理的单边沿时间数据包，根据 CH_ID 筛选用户所选的通道号，提取时间戳信息。

$$t = T \times 0.976\,ps$$

12.11 时间全局符合数据 2

表格 30 时间参考/全局符合数据格式说明

RT [63:60] ： 0xE	采集时间戳， LSB=10 ns [59:28]	CH_ID-A [27:23]:通道号	CH_ID-B [22:18]:通道号	Tdiff(TB-TA) [17:0]:时间差 LSB = 7.8 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。

$$t = Tdiff \times 7.8\ ps$$

.

相对于开始采集起点时刻，符合事件发生的时刻值 Tcoins 如下公式计算。

$$t_coins = Tstamp \times\ 10\ ns$$

注意时间差的正负与 12.2 节描述相同。

此类型符合数据可以精准定位符合事件发生的时刻，可以在多次周期性 Start-Stop 测试中根据时间戳值判断 Start-stop 发生事件在时间上的序列。

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 31 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		

14. 产品校准与维护

本产品几个关键参数可能会随着产品寿命变化，官方提供常规校准和维护服务。如果因为产品某些元器件过于老化造成的性能退化，官方提供有偿更换零部件服务。常见的产品维护参数以及保养建议如下表所示。

序号	参数	可能出现异常	建议校准周期	官方维护方式
1.	DAC 线性度漂移	阈值设置误差变大	6 个月	修正参数表
2.	设备温控晶振稳定性老化	计数率统计出现误差，长时间时间测量出现漂移	6 个月	原子钟校准+修正参数表
3.	数据接口芯片老化造成时序变化	接口带宽降低	12 个月	带宽测试+调整时序/更换芯片
4.	风扇老化	风扇噪声变大或转速变慢，设备温度变高	12 个月	风流量测试+风扇更换
5.	SMA 连接器磨损老化	时间测量精度下降，信号插损加大等	12 个月	信号幅度和前沿测试+连接器更换
6.	电源指示灯老化	指示灯亮度下降	12 个月	光强度测试+元器件更换

15. 常见问题答疑

本章我们针对用户反馈的问题以及可能遇到的问题进行答疑，方便用户迅速处理遇到的困难。

（1）问：我的应用场景没有空气流动，怎么才能保证设备散热？

答：对于没有空气流动散热的场景，您可以选择 Venus Lite 系列，而不是 Venus Lite-T 系列。Venus Lite 系列通过结构设计，将内部发热器件的热量引导到外壳上，然后用户可以把设备放在足够大的冷板上通过热传导进行散热。另外，Venus 会实时监控内部 FPGA 的核心温度，用户可以通过这个温度变化来检验传到散热的效果，如果能把 FPGA 的核心温度控制在 80 摄氏度之内，就不会对设备造成损伤，也不会影响时间测量分辨。

（2）问：我应该如何选择设备的电源适配器？

答：功耗要求：Venus Lite（-T）设备采用+12V 直流供电，设备功耗小于 24 W，因此需要保证电源适配器额定电流不低于 2 A。

电源接口物理尺寸：Venus 设备的电源接口触电内径为 2 mm，圆孔内径 5.5 mm。

（3）问：如何选择接入 SMA 接口的射频电缆？

答：用户可以参考以下建议选择射频电缆

（3.1）长度：探测器到 Venus 设备应该尽可能短，以减小信号前沿/幅度损失而影响时间测量精度，也减少外部干扰噪声的影响；

（3.2）特性阻抗：Venus 要求射频电缆特性阻抗为 50 欧，并且尽可能保证精确和稳定，特别是特性阻抗在测试环境温度范围内保证稳定性。用户需要关注电缆的 Return Loss 参数来评估信号完整性。另外，Venus 输入端已经端接了 50 欧电阻匹配，所以用户不需要额外再重复加匹配电阻；

（3.3）屏蔽参数：电缆会引入外部噪声，应该选择屏蔽参数优良的射频电缆以保证测试系统的信噪比性能；

（3.4）电缆带宽：用户应该先评估待测信号的前沿，并估算信号的膝频率，然后选择电缆的带宽要足够高以保证信号前沿不发生显著劣化。一般来讲，如果

电缆的带宽范围为 DC-18 GHz， 1dB Loss @ 18 GHz 就可以覆盖绝大部分应用场景；

（3.5）插损：电缆和连接器不可避免地引入插损。用户应该保证 Insertion Loss 足够小（比如<1 dB @ 目标频率）以保证信号完整性；

（3.6）推荐品牌：目前射频电缆工艺已经非常成熟，可选产品很多。有一些老牌知名厂商，如 Mini-circuits、TE Connectivity 等等。

（4）问：如何选择 USB 线缆？

答：用户可以参考以下建议选择 USB 线缆

（4.1）长度：为了保证 USB 传输稳定性，USB 线缆应该足够短，尽量控制在 3 米之内。如果必须要延长，可以使用主动式延长线（<10 米）或者光纤延长方案（>10 米）。此类产品著名品牌包括 Lindy、OWC 等；

（4.2）规格：USB 线缆需要满足 5Gbps 传输带宽，且外部有编织屏蔽网最佳；

（4.3）USB 线缆带来的地环效应：某些糟糕的测量环境和低噪声测量需求，USB 会将 Venus 设备与测试电脑地短接在一起，与探测器组成地环。参考最新手册的 7.1 节。地环会引入额外噪声。此时，我们建议用户使用 USB 隔离器（如 CESYS ISO-U30），通过磁屏蔽或者光屏蔽将测试电脑与 Venus 设备的地进行隔离。如果条件允许，使用光隔离最佳（如 LINDY 42708);

（5）问:如何选择千兆网线？

答：用户可以参考以下建议选择千兆网线

（5.1）长度：为了保证传输稳定性，千兆网线应该足够短，尽量控制在 20 米之内。如果要延长，可以通过网络交换机、中继器甚至是网线转光纤方案进行延长；

（4.2）规格：需要满足 1Gbps 传输带宽，符合 CAT6/7 标准；

（4.3）是否可以通过商用网络交换机：可以，但是如果交换机接入多台 Venus 设备组网时，应该保证每台设备的 IP 和 MAC 地址的唯一性。并且由于流量分担的问题，每台设备的带宽会打折扣。具体设置可参考 11.2.1 节；

（6）问:我需要使用 QSFP 接口接 4 路光纤到数据获取系统，通信协议怎么确定?

答：Venus 硬件上保留了 1 路 QSFP 接口，并且默认 1 路为万兆网络协议。用户需要使用 QSFP 转 SFP 模块（如 NVIDIA MAM1Q00A-QSA）来使用。如果用户需要使用 4 路光纤与自身的数据获取系统通信，需要联系时溯科技官方进行协议定制和协商。

（7）问:我购买的 16 通道测量系统，为什么还有 1 路参考通道（REF）？REF 通道有什么作用?

答：Venus Lite16 版本上有一路 REF 通道，这并不意味着 REF 通道在数据处理上有什么特殊之处。也就是说，REF 通道在数据处理上与其他 16 个通道完全一样，也是一路 6 ps RMS jitter 的 TDC 通道。唯一特殊的是，REF 通道前端没有比较器，用户需要输入一路 LVCMOS25 电平的数字脉冲信号进行测量。这样，用户可以利用这一路 REF 通道，在外界温度变化很大的场景下来标定比较器延迟的温漂。具体如下：

（7.1）做两端符合是不是必须接 REF：不是，Venus 采用任意通道全局符合算法，REF 通道并不特殊。参考 8.7.11 节；

（7.2）Venus Lite4/8 版本有没有 REF 通道：没有，只有 Venus Lite16 版本有 REF 通道；

（7.3）我恰好有一路数字脉冲信号需要测量时间，是否可以使用 REF 通道：可以，相当于数字脉冲直接接入 TDC 中测量。

（8）问:我想使用外部时钟作为 TDC 测量时钟，需要输入时钟的频率?

答：Venus Lite 设备上有 Clock input SMA 接口，需要输入一路时钟给 TDC 工作使用。对于 1.4+硬件版本，用户需要输入 10 MHz 或者 25 MHz 外部时钟（LVCMOS33/TTL 电平），且需要在软件上选择 10 MHz 或者 25 MHz 外部时钟选项。对于 1.3 硬件版本，仅支持 25 MHz 外部时钟（LVCMOS33/TTL 电平）。具体如下：

（8.1）TDC 如何 Reset：通道外部输入 Sync input SMA 脉冲（LVCMOS33/TTL

电平，脉宽超过 40 ns）来进行 TDC 时间戳值硬复位，或者，通过软件上的 TDC Reset 按钮进行 TDC 时间戳值软复位；

（8.2）设备之间级联的话，怎么处理：参考 11 章内容，遇到问题联系时溯科技；

（9）问:设备上有 Gate input 是做什么用的？

答：Gate input SMA 接口用于设备采集硬启动，也就是在其上输入用户指定的脉冲进行数据采集，用于超高精度（± 6.7 ns）的精确时间采集或多设备高精度同步数据采集。具体参考 8.6.8 节；

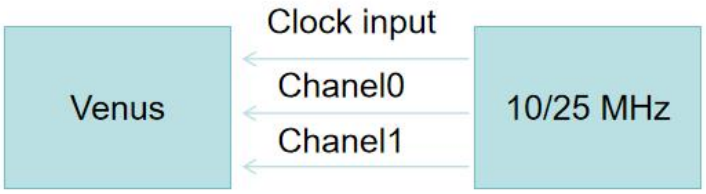
（10）问:我在高温工作环境下，设备风扇转的很快。

答：这个是正常的，Venus 设备使用温度反馈控制风扇转速方式，在高温环境下，风扇逐渐变快。如果不需要或者风扇影响了实验，请联系时溯科技。

（11）问:我使用外部 TDC 时钟，测量与 TDC 时钟同源的信号进行测量，会不会出现问题？

答：数据质量本身不会出错，但是用户会觉得数据不那么好看。我们结合测试进行解释和说明：

（11.1）本次测试平台如下：



10/25 MHz 时钟源完全相同地扇出输入到 TDC 外部时钟接口、测量通道 0 和测量通道 1 中。这种情况是采样时钟与测量信号完全相关的最差情形；

（11.2）这种情况下，TDC 总是测量几个固定细时间区域，而非常缓慢地变化。用户可以借助 Venus 地 GUI 软件上地 TDC 非线性分析功能测试 TDC 的 DNL 来观察这种相关性，如下。



（11.3）这种情况下并不是说明测量不对，而是输入信号的特殊性造成这种现象。虽然 FPGA 内部有细计数非线性修正算法，但是只能修正固定 BIN 附近一些码值，全局上还是非常有规律性。因此，这种情况下的时间分辨并不会受到显著影响，仅仅是分布直方图不那么光滑。如下所示，上图是外部 TDC 时钟和输入信号完全相关，下图是使用内部 TDC 时钟。上图看起来分布不高斯，但是拟合出来的时间分辨并不显著恶化（上下图拟合出来的时间晃动相同）。





（11.4）有没有办法避免这种情形：当然有，比如：

（11.4.1）TDC 时钟与测量信号不同源：绝对不会发生这种现象；

（11.4.2）TDC 时钟与测量信号同源，但是频率不恰好是整数倍，会相当程度地避免这种现象；

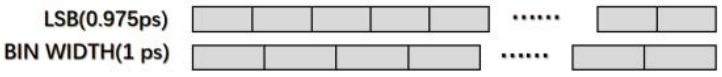
（11.4.2）TDC 时钟与测量信号同源，且频率恰好是整数倍，可以增大画直方图地 BIN WIDTH 来消除这种视觉效果，且并不显著影响性能；

再次重申，这种现象并不是不对，而是相关测量必然出现的一种现象，且这个现象不影响时间晃动性能。

（12）我采集了双通道时间差二进制/十六进制 Raw data，并按照 1 ps BIN 宽画出了时间差分布图，为什么有规律性的“异常散点”出来？

答：因为 Venus 的时间最小分辨为 0.976 ps，如果画直方图按照 1 ps 的 BIN 宽来画，这个微小的差异就会引入类似于“拍频效应”的“异常散点”。原理如下：

- 由于TDC LSB（最小刻度为0.975 ps），而画图使用1 ps为BIN宽
- 绘制直方图时出现“频差现象”，即如下所示·



$$(N+1) \times LSB = N \times BIN_WIDTH$$

计算得到理论上的拍频系数 $N = 39$，与测试结果完全一致

当 N 为 39 整数倍时，理论上 TDC 计数值与直方图渲染 BIN 宽值累加出现整数倍的偏差关系，导致相邻计数重复并入，导致 2 倍偏差。

您可以通过以下方法解决：

（12.1）对时间差数据增加 Dither，来消除这个“拍频效应”。Venus 默认的上位机软件就增加了 0.2 ps 的 Dither 消除了这个现象。具体的方法是给时间差数据增加 0-0.2 ps 的随机数进去，不显著影响时间分辨情况下消除这种现象；

（12.2）改变画图的 BIN 宽。

再次重申，这种现象并不是数据不对，而是不精准画直方图的一种必然结果。

（13）我要采集二进制/十六进制 Raw data 原始时间戳，有时间戳数据（12.1 节）、单边沿时间戳数据（12.10 节）和双边沿时间戳数据（12.7 节）。该如何选择？

答：这三种数据格式都是脉冲输入触发时间的原始时间戳数据，物理意义是相同的，主要区别如下：

（13.1）时间戳数据：两个触发事件各自 32 位时间戳数据，通过 Timezone 形式拼接在一起，以提高数据传输效率。其时间戳的边沿类型是用户通过用户界面配置确定；

（13.2）单边沿时间戳数据：一个触发事件 64 位时间戳数据，解析直接简单。其时间戳的边沿类型是用户通过用户界面配置确定；

（13.2）双边沿时间戳数据：一个触发事件 64 位时间戳数据，包含两个边沿指示位。用户不需要指定触发边沿；

当用户需要同时需要一个信号的前沿和后沿触发时间戳信息，用于事例筛选等分析，应该使用双边沿时间戳信息；

用户可以通过以下方法选择：

（13.3）当用户使用 USB（千兆网/万兆网）传输时，当预估计数率< 40（15/90）Mtags/s 时，且只需要一个时间边沿信息，建议用户选择单边沿时间戳数据；

（13.4）当用户使用 USB（千兆网/万兆网）传输时，当预估计数率> 40（15/90）Mtags/s 时，且只需要一个时间边沿信息，建议用户选择时间戳数据；

（13.5）当用户同时需要两个时间边沿信息，建议用户选择双边沿时间戳数据；

（14）Output Gate SMA 接口输出什么信号？

答：Output Gate SMA 接口是多功能输出接口，用户通过管理页面中进行配置，具体参考 8.9.2 节。

（15）设备使用 1 年了，没什么问题，但是最近发现输出计数率与输入计数率不一致了，甚至会出现输出计数率略微大于输入计数率的情况。

答：设备内部使用了高品质温补晶振来进行计时。但即使如此，晶振的老化带来性能的不确定变化依然不可避免。比如在进行计数率统计时，设备需要计算 1s 的时间。由于晶振老化，计时可能会逐渐出现误差。遇到这种情况，请及时与时溯科技官方联系，并按时进行设备维护（第 14 章）。时溯科技官方会帮助您更新修正参数，以保证长时间采集的稳定性。

（16）我需要做 N 重符合统计，需要多个策略组，我看上位机软件上有硬件或软件符合，这个应该怎么选择？

答：如 8.7.11 节所述，硬件实现 N 重符合具有低延迟、高效率、高性能的优势，如果用户只需要统计计数率结果，而不需要符合中间数据，那么建议采用硬件符合方式；软件 N 重符合高度依赖上位机电脑性能，且其延迟大，但其可以部署更多策略数和输出符合过程数据。所以给出如下建议：

（1）如果用户只需要 N 重符合计数率结果，且符合策略组数目不超过 40 个，建议使用硬件符合机制；

（2）如果用户上位机电脑性能较好，输入计数率较低（比如<10 Mtags/s total），需要部署超过 40 个策略组，或需要观测中间符合过程数据，那么可以选择软件符合机制。

（17）我需要做 N 重符合统计，需要多个策略组，选择硬件方式时，部署策略时间较长？

答：对于硬件符合方式，上位机软件需要将大量的符合策略参数下发给设备，因此部署策略时间较长。但是一旦部署完成后，设备统计响应就会很快。如果

部署 40 组硬件符合测量，大概的部署时间为 25 s(USB 接口)或 3s (千兆网口)。如果对部署速度有要求，可以选择千兆网口。

（18）我在进行时间差测量或者周期测量时，为什么自己通过 Raw 数据分析的结果与上位机测试的结果有一点差异（~0.06%）？

答：按照技术规格表中的说法，用户应该特别注意，时间测量最小 LSB 准确的值应该是 0.9765625 ps，即 1000/1024 ps。本设备配套的上位机软件均采用此精确值转换。如果用户需要精确离线分析数据，建议用户也使用 0.9765625 ps。为了简化文档编写，本文档近似写为 0.976 ps。

（19）我在进行 Start-Stop 分组测试中，为什么采集不到数据？而相同的两个信号，在双通道时间差分布中有正常的数据？

答：双通道时间差分布分析中，软件并不区分两路信号的“先来后到”，即简单输出两路信号的到达时间差信息，并不区分两个通道的属性。而在 Start-Stop 测试中，因为用户指定了 Start 通道与 Stop 通道，原则上在物理实验中，Start 信号必要先于 Stop 信号到达。因此，如果由于线缆等延迟造成 Start 信号晚于 Stop 信号到达，则设备无法采集到用户想要的合理数据。对于这种情况，用户应该首先评估两路信号的系统延迟（线缆、电路等）并在用户界面中的通道延迟配置中给予补偿。补偿的方法可参考附录 J。

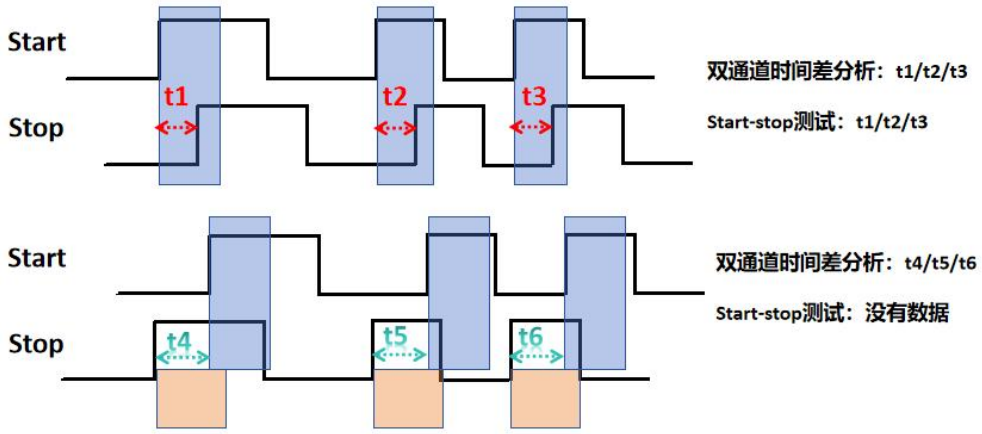


图 149 双通道时间差与 Start-stop 测试区别示意图（上升沿触发举例。

上图：Start 先于 Stop 信号到达，两种分析都测量到两个通道时间差 t1/t2/t3；

下图：Start 晚于 Stop 信号到达，双通道时间差分析不区分先来后到，因此直接输出两路信号的时间差 t4/t5/t6，而 Start-stop 分析，软件在 Start 信号到达后找 Stop 信号，因此没有正确的数据输出）

（20）我在进行 Start-Stop 分组测试中，Start 接探测器激发同步信号 Sync，Stop 接探测器输出信号 Signal。我想保存每一次 Sync 信号的 Sync 和 Signal 时间差，保存的数据能区分哪一次 Sync 激发？

答：完全可以。参考 8.7.3 节内容，你先进行 Start-stop 测试分组，并且勾选“保存 T3 数据”，软件将在 Output 文件夹下保存每一组，每一次 Sync 激发的信号与 Sync 信号时间差信息。这些信息足够你进行下一步的分析。

（21）我在进行 Raw 数据采集时，如何能检测到是否有数据丢失？

答：参考 12 章的数据帧格式中的 Frame cnt，可以通过 Frame cnt 的连续性判断采集过程中有无帧数据包丢失。

（22）输出的时间戳是否是按照时间先后顺序排列的？

答：是的。输出的时间戳数据是在 FPGA 中按照时间先后顺序排列的，排序长度为 4 倍的通道数。比如对于一个 16 通道的设备，排序长度为 64。即时间数据流进入一个 64 深度的排序池，每 64 个时间戳进行流水线实时排序。

（23）REF 通道舒是否可以直接接入数字信号？此通道的抖动和最高计数率是多少？

答：是的，REF 通道没有模拟前端电路，仅有过压保护电路。用户可以输入一个数字脉冲到 REF 通道进行时间测量，但需要保证信号在 50 欧姆电阻上的电压超过 2.7 V。REF 通道由一个 6 ps jitter 的 TDC 组成，计数率没有做特殊限制。

（24）TDC 外部输入参考时钟能否是正弦波？我想让秒脉冲 PPS 去复位 TDC，应该怎么做？

答：一般来说，建议输入方波。但是输入正弦波系统也可以工作，但是需要注意信号幅度要满足要求（在 50 欧姆电阻上电压幅度超过 2.7 V）。TDC 外部复位信号通过 Sync 输入 SMA 接口输入，只需要一个约 10+ ns 脉冲来复位 TDC。用户需要对 PPS 进行处理，把与 PPS 信号的上升沿同步的窄脉冲输入到 Sync 输入 SMA 中即可。

附录

A. 官方提供设备和附件列表

表格 32 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	电子版
3.	产品出厂质量检查报告	1	基础	
4.	电源适配器/线	1	基础	
5.	USB3.0 线缆	1	基础	
6.	千兆网线	1	基础	
7.	上位机软件安装包	1	基础	官网下载
8.	API 接口函数包	1	基础	
9.	MATLAB/Python 数据分析脚本	1	基础	
10.	万兆网卡+光纤转接器	1	选配	需额外付费
11.	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。随着产品继续迭代，我们可能会随时进行更新。如果用户需要固件升级，可以根据官方指导进行升级，也可以寄回公司进行免费升级。

表格 33 下位机固件版本说明

固件版本	升级内容	支持硬件
V9	1. 支持硬件在线 N-fold 符合(40 组）；	V1.3/V1.4/V1.5+
V10	1. TDC 非线性校准源采用真随机源，时间性能进一步提升； 2. 在 Raw 数据中增加了 Frame cnt，检测丢包； 3. 进一步提升 USB 连接稳定性。	

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 34 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/V1.3	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；
V1.4	2026.8	量产机版本	3. 调整 USB 接口带宽 4. 增加外部 10 MHz 时钟输入 5. 调整设备外观
V1.5	2026.9	量产机版本	与 V1.4 相同

表格 35 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	
V1.1.0	2026.4	配合 V1.3 硬件版本响应改动	
V1.2.0	2026.6	1. 增加 g2 计算； 2. 优化时钟频率分析计算； 3. 增加时间戳模式输出； 4. 增加 N 重符合统计分析； 5. 修改其他问题	
V1.3.0	2026.8	1. 优化了万兆网采集效率； 2. 增加了硬件 N 重符合； 3. 修改了其他问题	
V1.4.0	2026.9	1. 增加了 FLIM 扫描模块； 2. 增加了部分分析数据输出； 3. 优化了二维 Stop 热力图显示	
V1.5.0	2026.11	1. 开放计数率统计时间单位； 2. 优化了部分性能	

`

D. 对外数据接口列表

表格 36 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

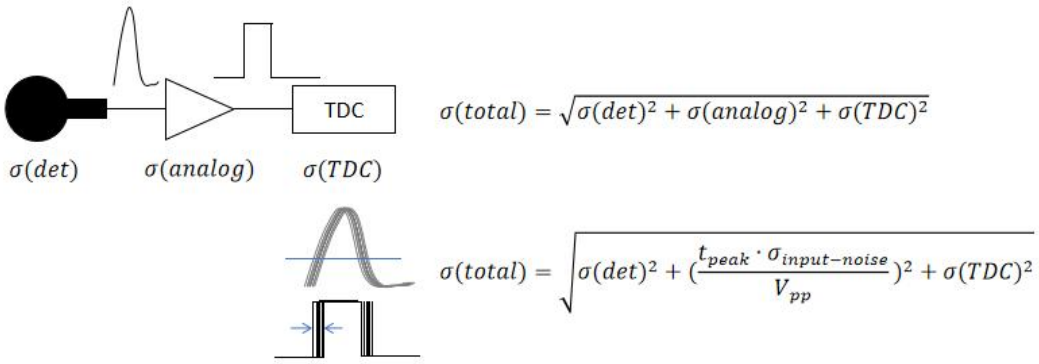


图 150 一般测量系统时间晃动分析模型

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 2 重符合逻辑说明

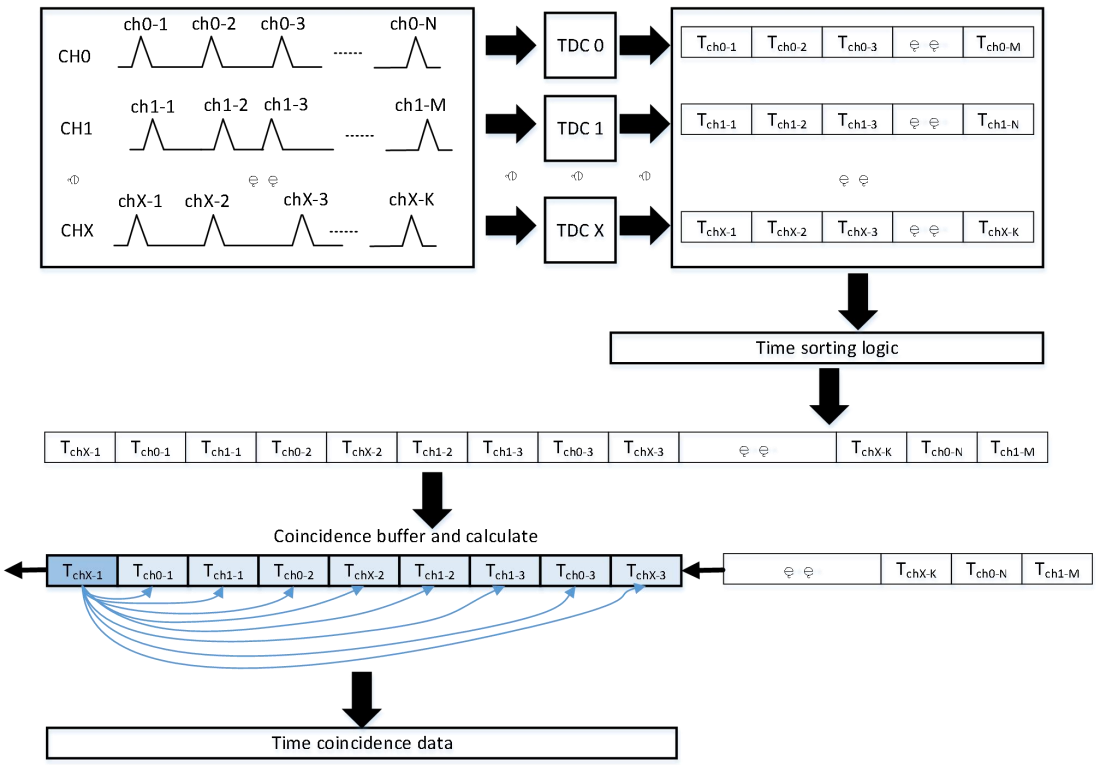


图 151 Venus 在线时间符合事件判断逻辑说明

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。

举例，如果通道 CH0、CH1、CH5、CH8 四个通道发生了符合事件，那么 Venus 将输出以下符合数据（比如本次事件按照以下时间先后发生：CH1-> CH5 -> CH0 -> CH8）：

Coincidence Raw data1: {CH1, CH5, Delt_T(CH1,CH5)};
Coincidence Raw data2:{CH1, CH0, Delt_T(CH1,CH0)};
Coincidence Raw data3:{CH1, CH8, Delt_T(CH1,CH8)};
Coincidence Raw data4:{CH5, CH0, Delt_T(CH5,CH0)};

Coincidence Raw data5:{CH5, CH8, Delt_T(CH5,CH8)};

Coincidence Raw data6:{CH0, CH8, Delt_T(CH0,CH8)};

因此，发生 N 重符合时，Venus 将输出$\frac{N(N-1)}{2}$个双重符合数据对。

G. 归一化滞后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 为时钟相位，τ为分析时间，N-2n 为平均次数，n 为时间常数τ内的采样周期数。

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

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

$$TDEV = \frac{\tau}{\sqrt{3}}MDEV$$

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

Venus 中归一化时间差滞后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。

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

H. 二阶自关联函数（g^2）算法

在量子光学中，二阶关联函数描述了光场强度在不同时刻的涨落相关性。根据 Siegert 关系，我们通过光子计数统计来估算：

$$g^2 = \frac{I(t)I(t+\tau)}{I(t)^2} \sim \frac{C(\tau)}{R^2 \cdot \Delta\tau \cdot T}$$

其中：

C(τ)：在时间延迟 τ 处检测到的光子符合计数（Coincidence Counts）。

R：平均计数率（Count Rate）。

Δτ：相关器时间通道的宽度（Bin Width）。

T：总采集时长。

该函数描述光子流的二阶统计特性，是 FCS 和 HBT 实验核心观测量。

I. 二阶互关联函数（g^2_{AB}）算法

双通道 g2 表征符合计数与随机计数的关系。如果完全理想的随机单光子源，归一化g^2_{AB}应该趋向于 1；如果两个通道相互关联，那么符合计数就大于随机计数。

设置采集的时间窗 T_win，即两个通道在这个时间范围内有效。假设光场的相干时间τ_C，符合门宽δ必须远远小于τ_C 本次测量才有意义。如果两个通道的相干时间非常短，那么无论如何测量都相干，也就失去了意义。因此δ<τ_C 是测量的前提。

此时，如果：

- 在延迟τ下，门宽δ时，光子符合计数值为 n(τ);
- γ_1 和γ_2 为 2 个探测器暗计数计数率;
- R_1 和 R_2 为 2 个探测器光子计数率;
- T 为采集时间;

$$g_{AB}(\tau)^2 = \frac{\mathrm{n}(\tau) - T\delta(R_1+\gamma_1)(R_2+\gamma_2) + T\delta R_1 R_2}{\mathrm{T}\delta R_1 R_2}$$

这是完整的公式。用户需要先测量/评估两个通道的暗计数率γ_1 和γ_2，然后进

行测量。如果忽略暗计数影响，那么不输入暗计数率即可。

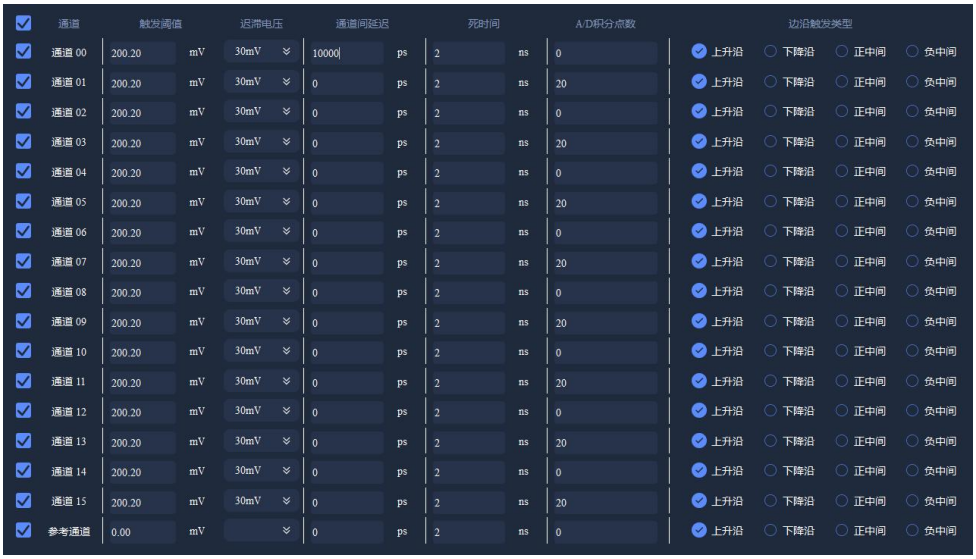
J. 通道间延迟值对齐方法

方法 1：最直接

由于每个通道延迟只能配置正值，所以所有通道必须与最大延迟对齐。最简单的办法：

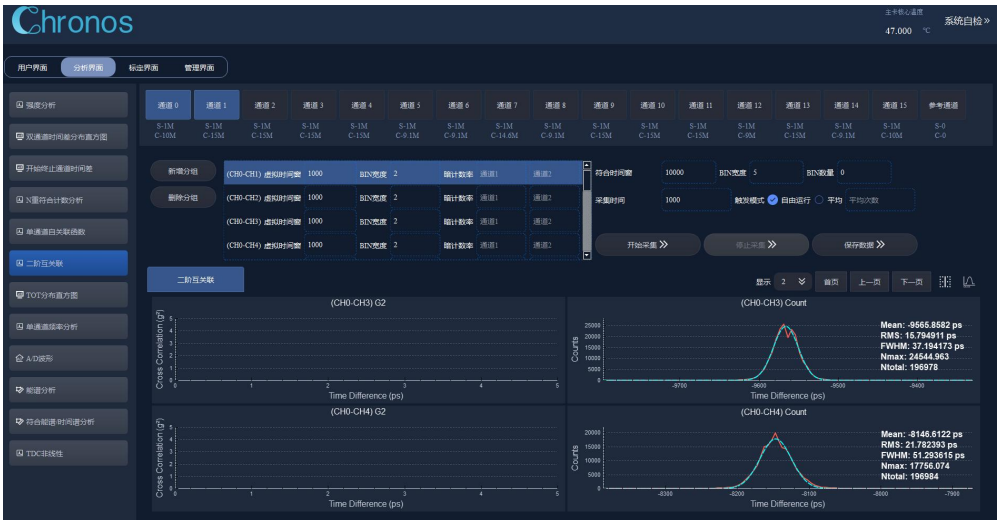
（1）评估系统各个通道最大时间延迟 Td_max；

（2）假设设置通道 0（CH0）延迟值为 Td_max。此时，CH0 已经成为了最大延迟通道，其他通道需要调节延迟值与之对齐；



通道	触发阈值		迟滞电压	通道间延迟		死时间		A/D积分点数	边沿触发类型
通道 00	200.20	mV	30mV	10000	ps	2	ns	0	上升沿 下降沿 正中间 负中间
通道 01	200.20	mV	30mV	0	ps	2	ns	20	上升沿 下降沿 正中间 负中间
通道 02	200.20	mV	30mV	0	ps	2	ns	0	上升沿 下降沿 正中间 负中间
通道 03	200.20	mV	30mV	0	ps	2	ns	20	上升沿 下降沿 正中间 负中间
通道 04	200.20	mV	30mV	0	ps	2	ns	0	上升沿 下降沿 正中间 负中间
通道 05	200.20	mV	30mV	0	ps	2	ns	20	上升沿 下降沿 正中间 负中间
通道 06	200.20	mV	30mV	0	ps	2	ns	0	上升沿 下降沿 正中间 负中间
通道 07	200.20	mV	30mV	0	ps	2	ns	20	上升沿 下降沿 正中间 负中间
通道 08	200.20	mV	30mV	0	ps	2	ns	0	上升沿 下降沿 正中间 负中间
通道 09	200.20	mV	30mV	0	ps	2	ns	20	上升沿 下降沿 正中间 负中间
通道 10	200.20	mV	30mV	0	ps	2	ns	0	上升沿 下降沿 正中间 负中间
通道 11	200.20	mV	30mV	0	ps	2	ns	20	上升沿 下降沿 正中间 负中间
通道 12	200.20	mV	30mV	0	ps	2	ns	0	上升沿 下降沿 正中间 负中间
通道 13	200.20	mV	30mV	0	ps	2	ns	20	上升沿 下降沿 正中间 负中间
通道 14	200.20	mV	30mV	0	ps	2	ns	0	上升沿 下降沿 正中间 负中间
通道 15	200.20	mV	30mV	0	ps	2	ns	20	上升沿 下降沿 正中间 负中间
参考通道	0.00	mV		0	ps	2	ns	0	上升沿 下降沿 正中间 负中间

（3）进入分析页面的“二阶互关联”分析，建立所有通道与 CH0 的采集分组，即 CH0-CH1,CH0-CH2... 。然后点击“开始采集”，得到每个通道与 CH0 的时间差的分布。拟合得到延迟值平均值 MEAN(CH_i)；注意：由于系统吞吐量限制，要求每个通道输入信号数据率不要超过 0.6 Mtags/s；



（4）回到用户页面，把每个通道的延迟配置成-MEAN(CH_i)即可。理论上这样所有通道就和 CH0 延迟相同了。注意：由于拟合精度，外界环境变化等因素，对齐可能会存在几个 ps 的误差，请根据实际测试评估。



方法 2：使用时溯科技提供的基于 Matlab 的参考分析程序，自动生成各个通道的延迟值。参考 9.6 节。

（1）首先在用户界面，选择“全局符合数据”模式，采集全局符合 raw 数据；

（2）打开 Venus_Raw_data_analysis_vx.x.x.m 程序，读入采集到的 Raw 数据；

（3）生成 channel_delay_lut 数据，第一列为通道号，第三列为对齐需要配置的延迟值（ps）；

（4）进入用户界面，配置延迟值。

本文档历史记录

文档编号	修改时间	修改人	说明
UG-01-3-010-1	2025.12	Cong	V1.0
UG-01-3-030-1	2026.2	Cong	1. 针对 v1.2 硬件版本进行更新； 2. 优化了时间测量性能； 3. 增加了比较器迟滞电压设置功能； 4. 去掉了时间延迟使能按钮； 5. 修改了外部时钟输入要求； 6. 增加了 LabVIEW SDK，更新了 C++/Python/Matlab SDK； 7. 增加了 Start-Stop 分析模式； 8. 更新了部分测试结果
UG-01-3-040-1	2026.4	Cong	1. 针对 v1.3 硬件版本进行更新； 2. 详细描述了 Start-stop 统计方法； 3. 更新了产品订购型号说明； 4. 更新了测试结果和 API 接口
UG-01-3-050-1	2026.6	Cong	1. 增加了关联函数 g2 计算； 2. 优化了时钟频率分析功能，满足 IEEE 1139； 3. 增加了多重符合统计； 4. 解决了部分 bug；
UG-01-3-051-1	2026.7	Cong	1. 增加了单边沿数据格式； 2. 增加了多功能 Gate 输出； 3. 增加了常用问题答疑
UG-01-3-052-1	2026.8	Cong	1. 增加了硬件 N 重符合统计方案； 2. 增加了硬件 N 重符合统计性能测试结果；
UG-01-3-053-1	2026.9	Cong	1. 增加了 FLIM 模块描述； 2. 更新了部分测试结果

UG-01-3-054-1	2026.11	Cong	1. 增加了 Burst 测试； 2. 增加了组网 USB 采集方法； 3. 增加了 Frame cnt 数据结构； 4. 更新了部分测试结果 5. 增加了一些答疑