Chronos

ALEA16
1.6 Gbps, Research-Grade
True Random Number Generator

ALEA16 User Guide

UG-03-1-010-1, 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 7
2. 技术规格 8
3. 产品订购选型 10
4. 产品功能结构和应用场景 10
5. 产品外观尺寸 11
6. 绝对最大额定值 11
7. 设备接地和静电 11
8. 上位机软件使用说明 12
    8.1 软件安装说明 12
        8.1.1 Windows 系统 12
        8.1.2 Linux 系统 18
    8.2 上位机电脑硬件要求 20
    8.3 接口配置 21
        8.3.1 千兆网口 21
        8.3.2 USB 3.0 23
    8.4 界面介绍 23
    8.5 登录页面 24
    8.6 用户页面 25
        8.6.1 随机数计数率选择 25
        8.6.8 采集模式选择 26
        8.6.9 测量数据保存 27
    8.7 分析页面 28
        8.7.1 强度分析 28
        8.7.2 香农熵分析 28
    8.8 管理页面 30
        8.8.1 基本介绍 30
    8.9 日志区 31
9. 开发者 API 接口 32
    9.1 C API 32
        9.1.1 概述 32
        9.1.2 初始化和销毁 32
        9.1.3 设备连接 32
        9.1.4 错误处理 34
        9.1.5 设备自检 36
        9.1.6 通道配置 36
        9.1.7 全局配置 37

9.1.8 状态查询 38
9.1.9 网络配置 40
9.1.10 数据采集 41
9.1.11 数据分析 42
9.1.12 其他辅助接口 43
9.1.13 通用注意事项 45
9.2 C++ API 45
9.2.1 概述 45
9.2.2 初始化与设备连接 45
9.2.3 设备自检 48
9.2.4 通道配置 48
9.2.5 全局配置 49
9.2.6 网络配置 51
9.2.7 数据采集 53
9.2.8 数据分析 55
9.2.9 注意事项 56
9.2.10 示例代码 56
9.3 PYTHON API 58
9.3.1 概述 58
9.3.2 初始化 58
9.3.3 设备连接 58
9.3.4 错误处理 59
9.3.5 设备自检 60
9.3.6 通道配置 60
9.3.7 全局配置 61
9.3.8 状态查询 61
9.3.9 网络配置 63
9.3.10 数据采集 64
9.3.11 数据分析 65
9.3.12 注意事项 66
9.3.13 示例代码 67
9.4 MATLAB API 69
9.4.1 概述 69
9.4.2 初始化 69
9.4.3 设备连接 69
9.4.4 错误处理 70
9.4.5 设备自检 71
9.4.6 通道配置 71

9.4.7 全局配置 71
9.4.8 状态查询 72
9.4.9 网络配置 73
9.4.10 数据采集 74
9.4.11 数据分析 75
9.4.12 注意事项 76
9.4.13 示例代码 77
9.5 LabVIEW API 82
9.5.1 概述 82
9.5.2 环境准备 82
9.5.3 初始化和销毁 82
9.5.4 设备连接 83
9.5.5 设备自检 84
9.5.6 通道配置 85
9.5.7 全局配置 86
9.5.8 状态查询 87
9.5.9 网络配置 90
9.5.10 数据采集 92
9.5.11 数据分析 95
9.5.12 注意事项 96
10. 典型测试电路和测试结果 97
10.1 测试简介 97
10.2 RO 周期测试 97
10.3 熵源输出测试 98
10.3.1 NIST SP 800-90B 测试 98
10.3.2 NIST SP 800-22 测试 101
10.3.3 Dieharder 测试 102
10.3.4 AIS-31 测试 103
10.3.5 分布均一性测试 104
10.3.6 温度漂移测试 105
10.3.7 重启测试 108
10.4 SM3 输出测试 108
10.4.1 NIST SP 800-90B 测试 108
10.4.2 NIST SP 800-22 测试 109
10.4.3 Dieharder 测试 110
10.4.4 AIS-31 测试 111
10.4.5 分布均一性测试 111
10.4.6 温度漂移测试 112

10.4.7 重启测试 113
11. 测量类型原始数据结构 114
12. 支持定制化和二次开发 115
13. 产品校准与维护 115
14. 常见问题答疑 116
附录 117
A. 官方提供设备和附件列表 117
B. 固件版本说明 117
C. 硬件和软件版本说明 118
D. 对外数据接口列表 118
E. RCT 和 APT 计算公式 119
本文档历史记录 120

1. 产品特征

功能丰富



- 最高 1.6 Gbps 速率真随机数生成器
- 用户可选输出速率 100 Mbps- 1.6 Gbps
- 用户可选熵源模式和 SM3-GM/T 0004 后处理模式输出
- 集成 Startup/RCT/APT 熵源健康监测
- USB3.0 (2.5 Gbps)/1G 以太网(900 Mbps)/10 G 以太网(9 Gbps)数据接口
- 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

- 集成 16 通道 3 ps jitter 高性能 TDC 模块
- 提取 FPGA 内部时钟抖动熵，熵源物理过程清晰，无额外动力学
- 全数字硬件实现
- 支持熵源模式输出
- -10 ~ +40 ℃ 工作环境温度下 IID 最小熵大于 0.99 bit
- 输出随机数经过 NIST-SP-800-22/NIST SP 800-90B/AIS-31/Dieharder 测试
- 经过重启测试

其他

- 可订制化
- 终身技术支持

2. 技术规格

+12 V 直流供电。

表格 1 ALEA16 系列系统技术规格参数

参数	ALEA 16	单位	测试条件/说明
建议工作环境温度	-10/40	℃ min/max	
系统散热	风冷		
风扇个数	1		风扇自动调速
风扇转速	4300	RPM	转速实时监控
气流	9.7 x 1	CFM	温度实时监控
噪声	22 x 1	dB	
内置 TDC 通道数			
时间测量通道数	16	个	
熵源	时钟抖动白噪声		
TRNG AUX 输出			
输入信号高电平	2.4/3.3	Vmin/max	可定制输出
输入信号低电平	0/0.8	Vmin/max	
输出数据类型			
熵源数据	支持, >1.6 Gbps		
SM3 后处理数据	支持, >1.4 Gbps		用户可以采集/保存任一数据类型的裸数据，支持.bin/.hex 格式
SM3 标准	GM/T 0004		
SM3 压缩比	83.3 %		
健康自检			
RCT 连续 0 或 1 上限	23		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
APT 比特 1 概率上/下限	57.4/42.6	%	
启动监测样本数	10000		
对外数据接口带宽			
USB 3.0	2.5	Gbps	
千兆以太网	0.9	Gbps	
万兆以太网	9.0	Gbps	
系统供电			
直流电压/电流	+12/>3	V/A	
直流供电纹波要求	<0.1	Vpp	
系统功耗	<25	W	

设备尺寸	150x130x73	WxLxH,mm^3	
外壳材料	铝 6061		
丝印油墨材料	嘉宝莉 72		

3. 产品订购选型

表格 2 ALEA 系列产品订购选型

产品系列	产品	真随机数输出速率	对外接口	供电方式
ALEA	ALEA16	1.6 Gbps（熵源模式） 1.4 Gbps（SM3 模式）	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；

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

4. 产品功能结构和应用场景

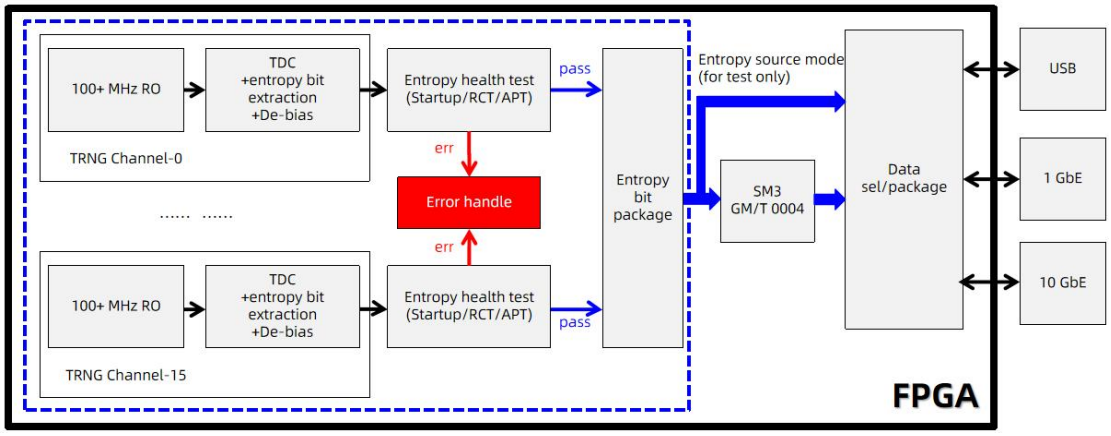


图 1 ALEA16 系列产品功能结构框图

ALEA16 采取多通道单比特熵提取架构，每个通道集成 1 个 Ring Oscillator（RO），产生约 110 MHz 的自由运行时钟，然后经过 1 个 3 ps jitter 高性能 Time-Digital Converter（TDC）进行单比特熵提取。用户可以直接使用产生的比特熵（熵源模式）或经过 SM3-GM/T 0004 后处理输出。用户可以选择 1 GbE/USB/10 GbE 三种数据接口进行数据采集。

有几个特性：

（1）用户可以关闭 16 个通道中的任意 1 个；

（2）自带熵比特健康测试，包括启动监测 Startup，重复计数检测 RCT 和自适应比例检测 APT，如附件 1 所述；

（3）用户可选择熵源模式输出或经过 SM3 后处理的数据输出。

5. 产品外观尺寸

6. 绝对最大额定值

表格 3 ALEA 设备工作额定值

参数	额定值	说明
电子		
直流供电	+11 V ~ +15 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	-10 ℃ ~ +40 ℃	

7. 设备接地和静电

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

8. 上位机软件使用说明

8.1 软件安装说明

官方提供 ALEA 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”。

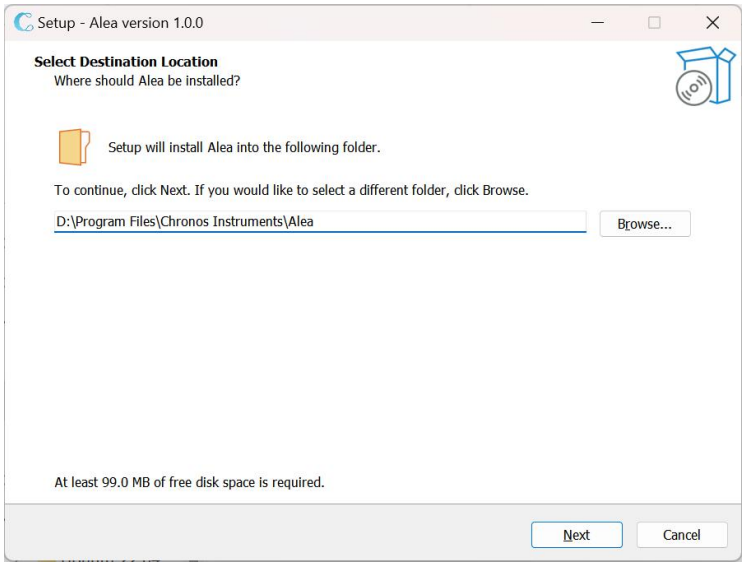


图 2 上位机安装步骤 2

（3）选择是否创建桌面快捷方式

点击“下一步”；

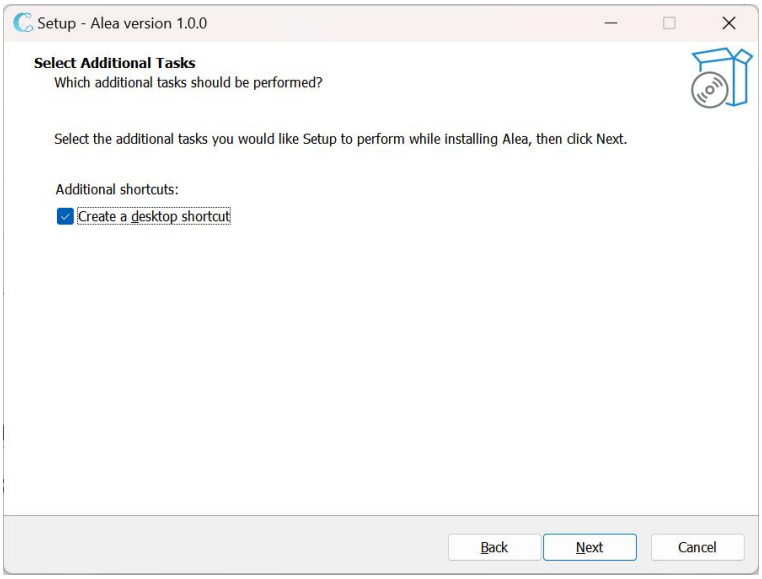


图 3 上位机安装步骤 3

（4）开始安装

点击“install”，将开始安装过程；

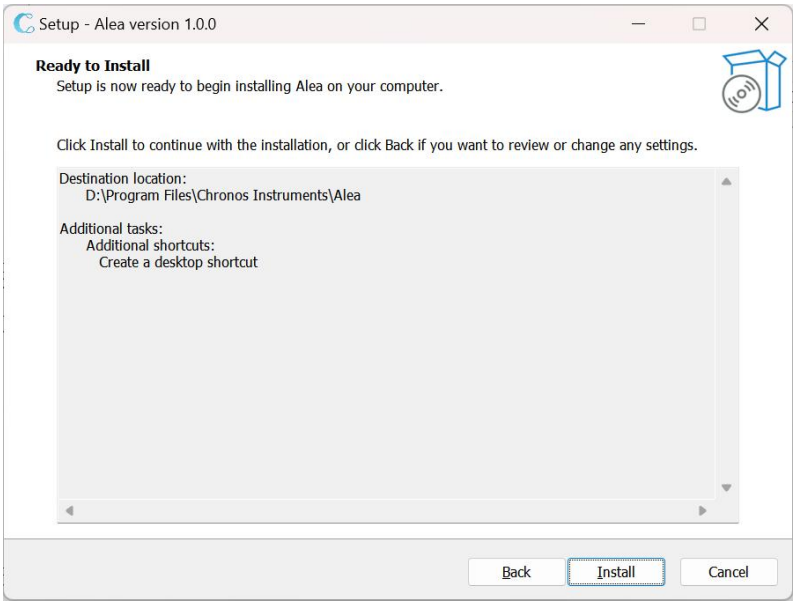


图 4 上位机安装步骤 4

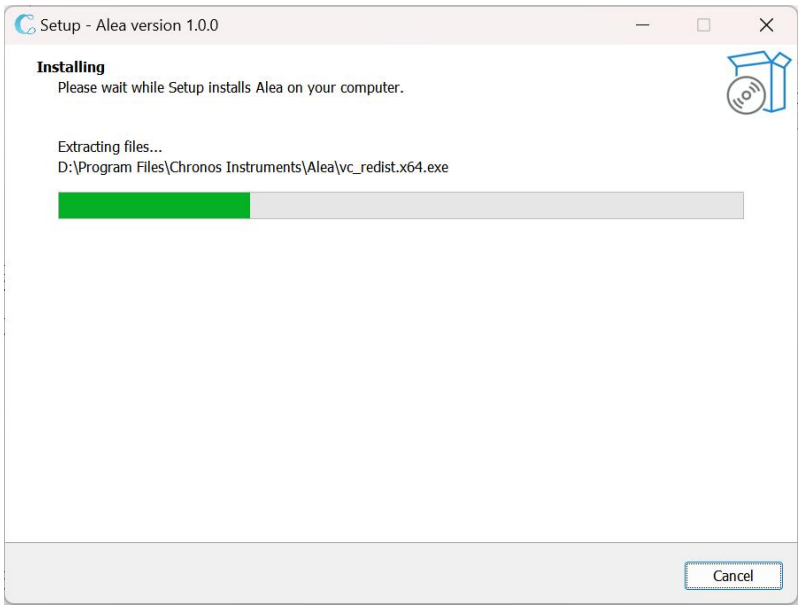


图 5 上位机安装步骤 4 安装过程

（5）安装成功

如果安装成功，将显示如下界面。

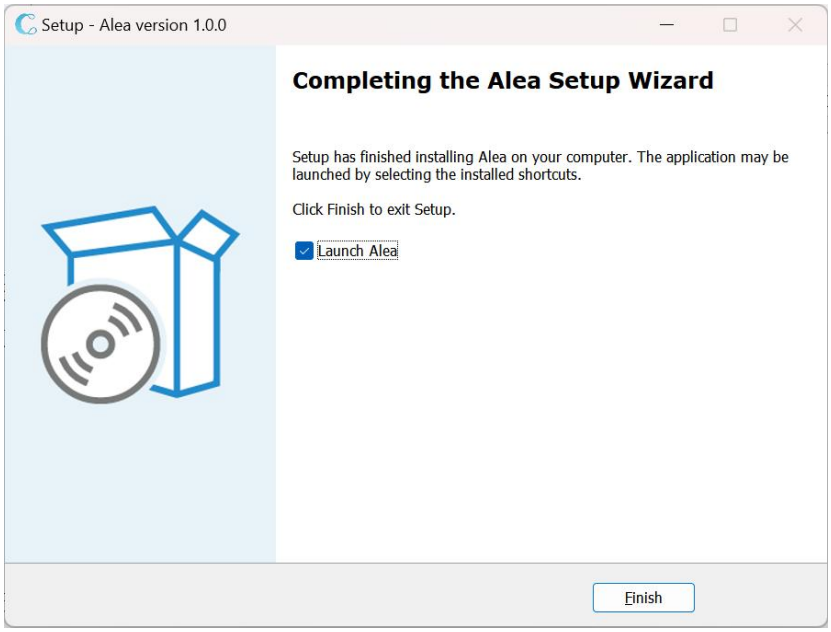


图 6 上位机安装完成

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。

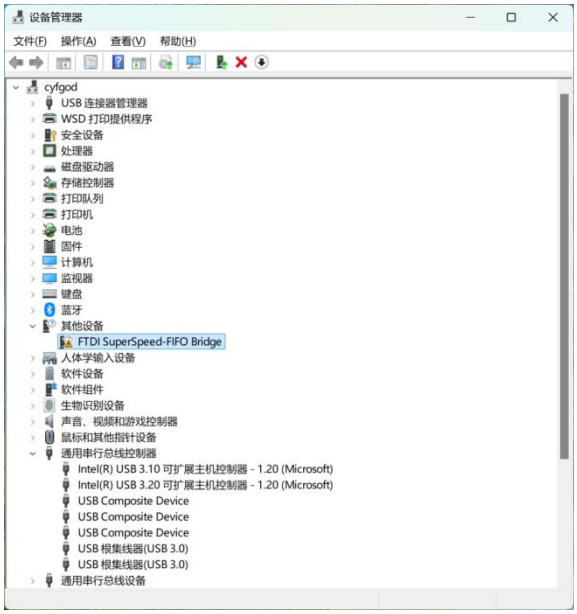


图 7 USB3.0 驱动设备管理器

（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 8 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 9 手动安装 VC++运行库

8.1.2 Linux 系统

8.1.2.1 软件安装步骤

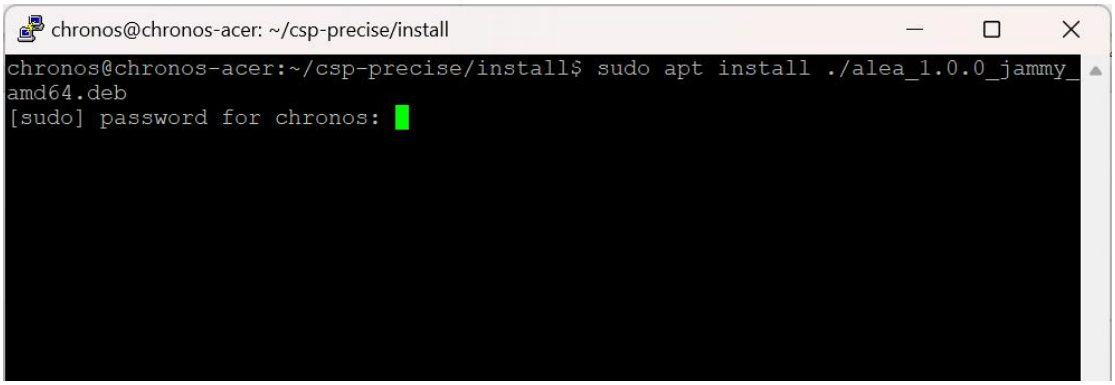
（1） 确认软件包

目前的 Linux 软件包主要以 Deb 包格式分发：

Deb 安装包文件：ALEA_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 ./ALEA_1.1.0_jammy_amd64.deb



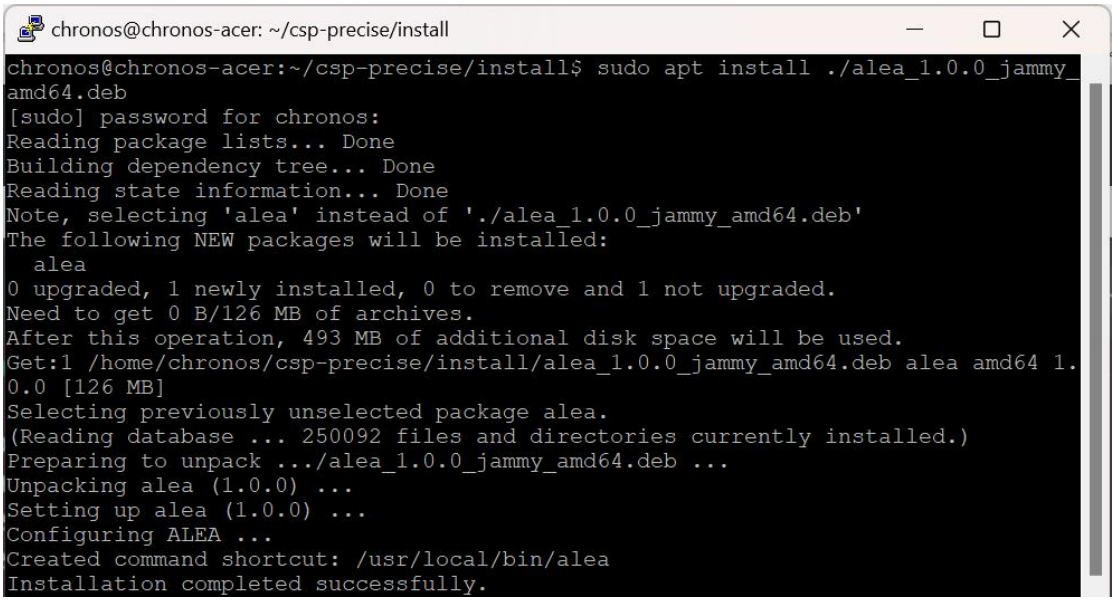


图 10 软件安装 Linux 命令

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/ALEA 目录下，可以直接输入 ALEA 运行程序。

首先是会弹接口选择页面，选择设备接口。

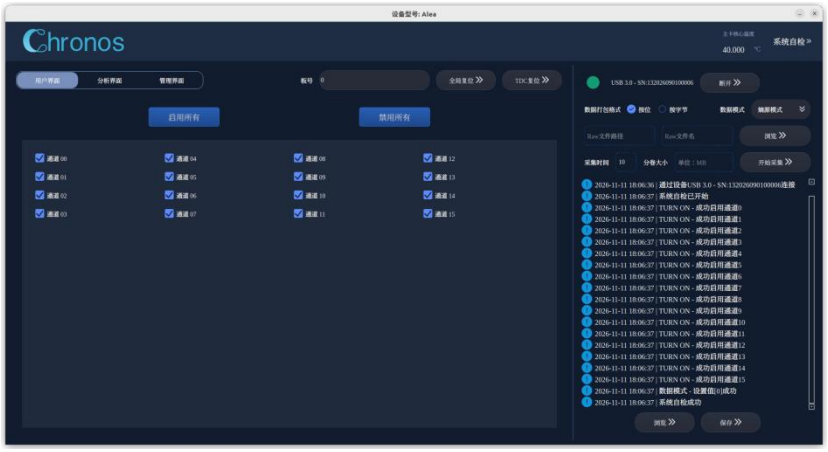


图 11 Linux 系统下的软件运行

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 3GHz 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

ALEA 提供二种对外数据接口：千兆网口和 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

ALEA 设备出厂默认配置如下：

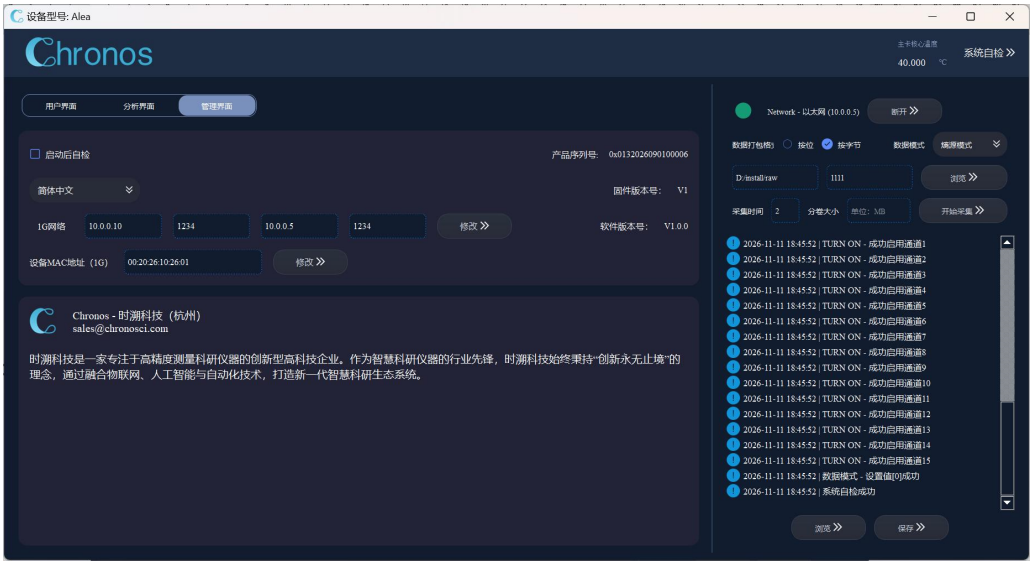
- 设备本地 IP 地址: 10.0.0.10
- 上位机电脑 IP 地址: 10.0.0.5
- 设备本地 MAC 地址: 12-34-56-78-90-AB
- 通信端口号: 1234 (设备与上位机默认使用相同端口)

请按以下步骤建立千兆网连接：

1) 使用千兆网线将 ALEA 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) ALEA 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/aleaconfig.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 12 上位机电脑千兆网网络设置



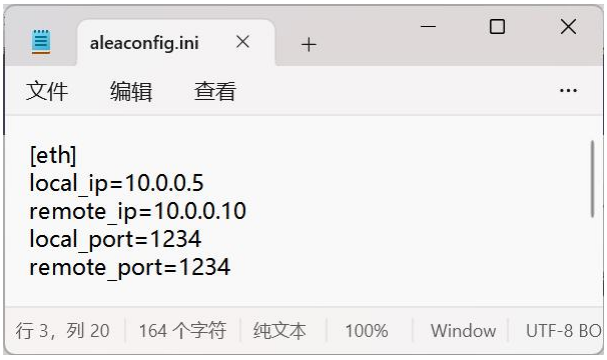


图 13 网口 IP 和端口地址修改页面

8.3.2 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 ALEA 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 14 USB3.0 驱动安装信息

8.4 界面介绍

（1） ALEA 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 和千兆网接口的连接状态，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、设备以太网地址配置等。

8.5 登录页面



图 15 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 和 UDP1G (千兆网)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

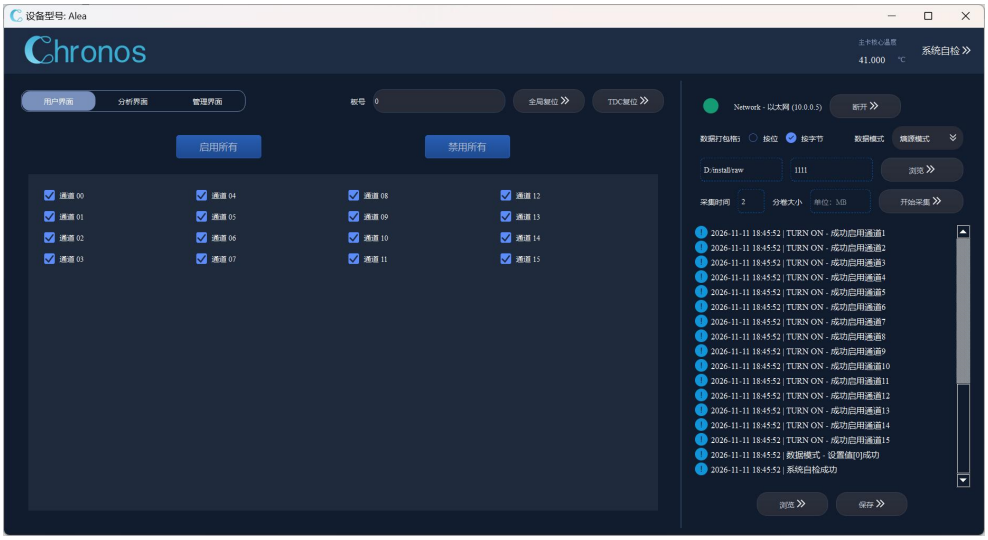


图 16 用户页面

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；

（2.2）系统配置：数据采集模式设置等。

（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 随机数计数率选择

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。如果用户全部选择，将输出 1.6 Gbps 随机数。如果只选择 1

路，则输出 ~100 Mbps 随机数，以此类推。



图 17 通道使能勾选

8.6.8 采集模式选择

1. 选择采集模式

根据测量需求，从以下模式中选择其一：

- 熵源模式
- SM3 模式

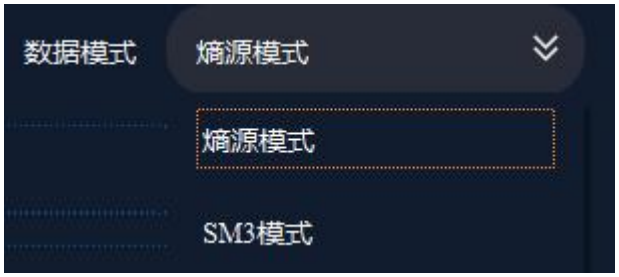


图 18 数据采集模式选择

数据采集模式解释如下：

（1） 熵源模式：在此模式下，ALEA 输出无任何后处理的熵源比特流；

（2） SM3 模式：在此模式下，ALEA 输出熵源数据经过 SM3-GM/T 0004 算法后处理的熵源比特流。

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	说明
熵源模式	Entropy Source Mode	所有通道	ALEA 输出无任何后处理的熵源比特流
SM3 模式	SM3 Mode	所有通道	ALEA 输出熵源数据经过-SM3-GM/T 0004 算法后处理的熵源比特流

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 19 测量数据保存配置



图 20 数据打包格式

数据打包格式分为按位和按字节二种方式，默认都保存为.bin 格式。

（1）按位：随机数（0 和 1）按位直接保存；

（2）按字节：随机数（0 和 1）按字节保存；

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 ALEA 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

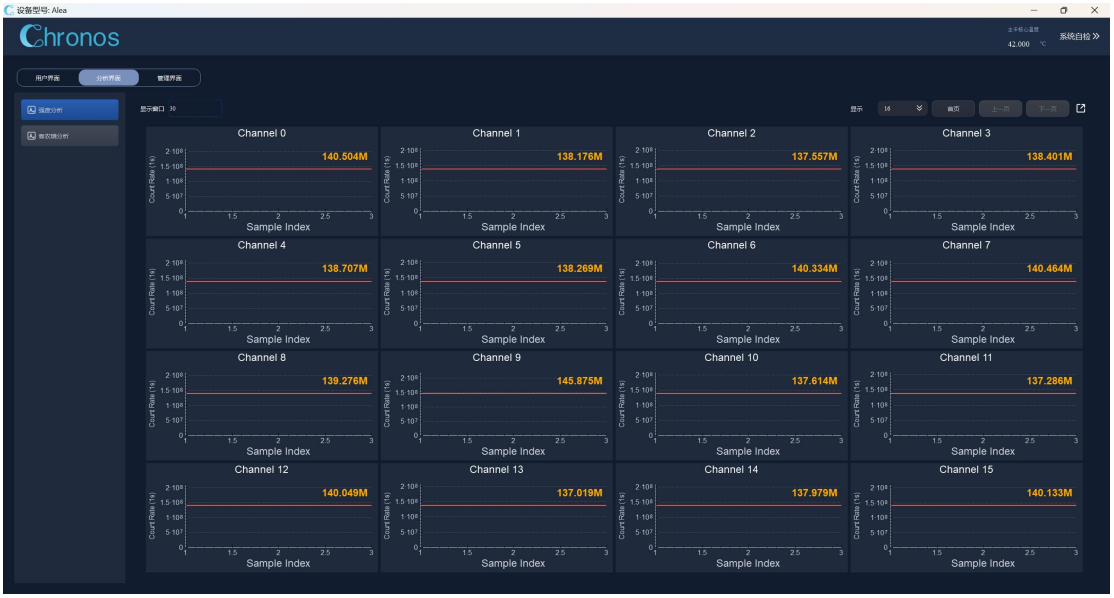


图 21 各通道强度（计数率）分析（实时计数率）

如上图所示，用户可进入“强度分析”功能页， ALEA 会开始工作，并以每秒一次的频率刷新和显示所有通道的实时计数率（Singles Counts Per Second，SCPS）。此窗口用户可以单独游离出来以随时实施观测。此功能为系统调试和性能评估提供了关键工具。

8.7.2 香农熵分析

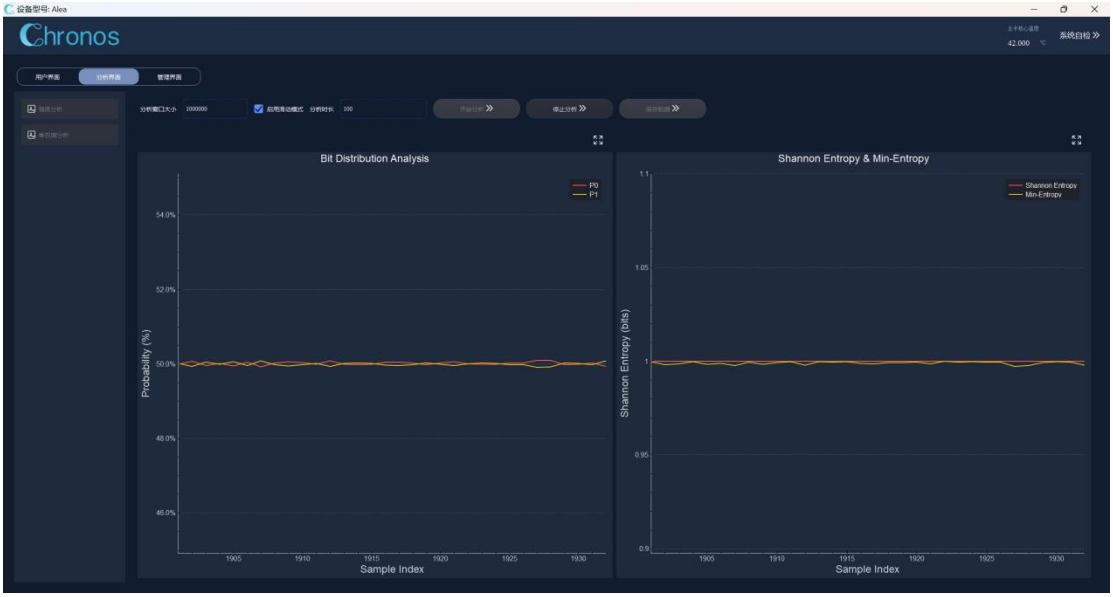


图 22 香农熵和最小熵分析

如上图所示，用户可进入“香农熵分析”功能页进行设备随机性能分析。用户可以指定分析窗口大小（至少要大于 100000），指定分析时长，点击“开始分析”按钮进行数据分析。

左边图表，是显示在每次采样（分析窗口大小内）的随机数 0 和 1 出现的概率，分布表示为 P0 和 P1 的分布曲线。

右边图表，是计算每次采样的香农熵和最小熵的值。随机性能越好，值越接近 1。

8.8 管理页面

8.8.1 基本介绍

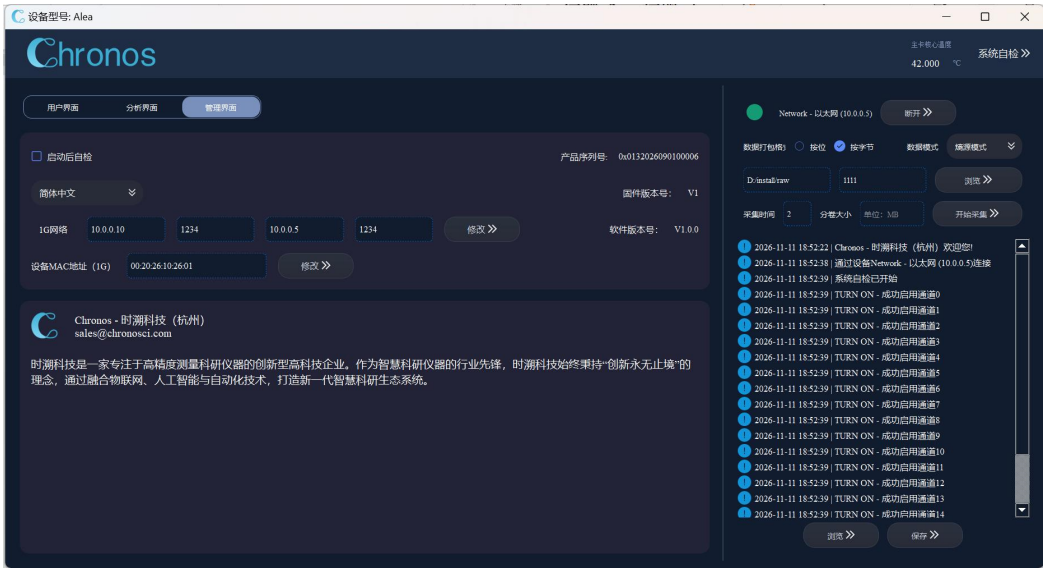


图 22 管理页面

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
- 以太网地址配置
- 设备序列号显示
- 下位机固件版本号显示

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

表格 6 设备序列号解释

设备序列号段	段名	可选
段 1	设备类型	0x3 ALEA(TRNG)
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 日志区

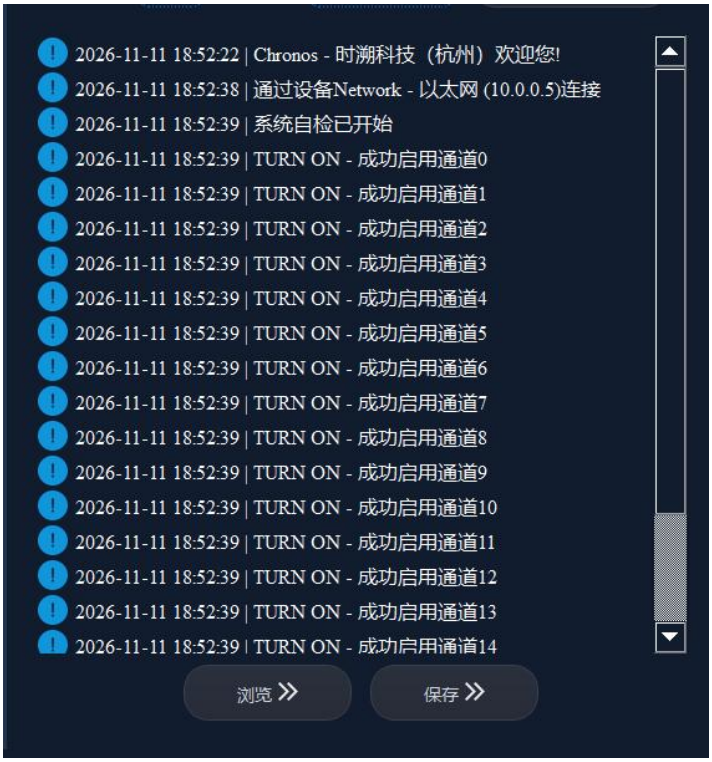


图 23 系统日志

ALEA 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：ALEA 真随机数生成器设备）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（暂不支持）

【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	设备信息包含类型和名称 支持两段式获取，第一次传入 NULL 数值，返回数组大小；第二次返回实际数组值
ci_get_device_num(ci_handle_t handle)	handle：对象句柄	int-设备接口数量	获取可用设备接口数量	
ci_get_device（ci_handle_t handle, int32_t index, int32_t*	handle：对象句柄 Index：设备 index out_type:设备接口	int - 状态码	获取指定设备信息（包括：接口类型、名	仅 USB3 接口类型支持设备序列号

out_type, char* out_name, int32_t name_size, char* out_sn, int32_t sn_size)	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		称和设备接口序列号）	
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	-	记录 ERROR 级别日志	

	message：日志信息			
ci_critical(ci_handle_t handle, const char* message)	handle：对象句柄 message：日志信息	-	记录 CRITICAL 级别日志	

错误回调函数定义：

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

9.1.5 设备自检

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

【API 定义】

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

9.1.6 通道配置

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
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_is_channel_enabled(ci_handle_t handle, int32_t	handle：对象句柄 channel：通道号	int - 状态码	获取通道使能状态	

channel, int8_t* out_value)	out_value:返回值指针			

9.1.7 全局配置

注意事项：

- 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；

【数据模式定义-cinst_alea_data_mode_t】

定义	值	说明
CINST_DATA_MODE_ENTROGY_SOURCE	0	熵源模式
CINST_DATA_MODE_SM3	2	SM3 模式

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
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	详见上面数据模式定义
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_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_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_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

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

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

9.1.9 网络配置

注意事项：

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

【API 定义】

API 函数	参数	返回值	功能描述
ci_set_ips(ci_handle_t handle, const char* local_ip, const char* device_ip)	handle：对象句柄 local_ip：本地 IP device_ip：设备 IP	int - 状态码	设置千兆网口 IP 地址
ci_set_net_ports(ci_handle_t handle, int32_t local_port, int32_t device_port)	handle：对象句柄 local_port：本地端口 device_port：设备端口	int - 状态码	设置千兆网口端口号
ci_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,	handle：对象句柄	int - 状态码	获取千兆网口端口号

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

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；
- 支持二种数据打包格式，可以根据实际需要，选择合适的格式类型；按位打包格式，数据最紧凑，数据采集效率最高。

【数据打包格式定义-cinst_data_packing_format_t】

定义	值	说明
CINST_DATA_PACKING_BITWISE	0	按位
CINST_DATA_PACKING_BYTEWISE	1	按字节

【API 定义】

API 函数	参数	返回值	说明
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)	int - 状态码	开始进行数据采集（异步模式） <data_mode> 请详见前面的 AleaDataMode 定义 <duration> 采集时间：单位-秒 <file_name> 未明确后缀名情况下，默认后缀名.bin； <max_volumesize_mb> 如果分卷大小小于等于 0，文件不分卷
ci_fetch_rawdata(ci_handle_t handle, int32_t* out_size, int32_t timeout_ms)	handle：对象句柄 out_size:返回数组大小 timeout_ms：超时时间（单位：ms）	char*	实时读取 Raw 数据 返回 char 数组
ci_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

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。

【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	
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	

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, const int64_t* data, int32_t data_points_size)	dest：目的内存指针 dest_size：目的内存大小 data：源数据地址 data_points_size：源数据组大小	int-实际字节数	转换 int64 数据到字符串	Int64 是数据分析 API 返回的一维数据，每 2 位数字表示一组 pair 数据。
ci_double_pairs_to_string(char* dest, int32_t dest_size, const double* data, int32_t data_points_size)	dest：目的内存指针 dest_size：目的内存大小 data：源数据地址 data_points_size	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。

	：源数据组大小			

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：ALEA 真随机数生成设备）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（暂不支持）

【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

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 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 数据模式设置

【数据模式-INST_ALEA_DATA_MODE】

定义	值	说明
INST_DATA_MODE_ENTROGY_SOURCE	0	熵源模式
INST_DATA_MODE_SM3	2	SM3 模式

【API 定义】

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

			nt> status - 状态 value - 数据模式类型值

9.2.5.3 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	
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, const std::string& device_ip)	local_ip	本地 IP 地址	设置和获取千兆网 IP 地址 返回 INST_NET_RESULT<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> 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 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

9.2.7 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；
- 支持二种数据打包格式，可以根据实际需要，选择合适的格式类型；按位打包格式，数据最紧凑，数据采集效率最高。

【数据打包格式类型-INST_DATA_PACKING_FORMAT】

定义	值	说明
INST_DATA_PACKING_BITWISE	0	按位打包
INST_DATA_PACKING_BYTEWISE	1	按字节打包

【API 定义】

API	参数	参数说明	备注
bool setDataPackingFormat(int dataPackingFormat)	dataPackingFormat	数据打包格式类型（详见前面数据打包格式定义）	设置数据打包格式
int getDataPackingFormat()	-	-	返回数据打包格式类型
bool acqRawData(int dataMode, int duration，const std::string& folderPath = "" , const std::string& fileName = "", int maxVolumeSizeMB = -1, bool forceClose = true)	dataMode	数据模式（详见前面的数据模式定义）	开始数据采集（异步模式，立即返回） 返回成功或失败。 如果设置了输出文件名，会启用数据文件保存。未明确后缀名情况下，默认后缀名.bin
	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）	
	fileName	输出文件名称（如果不指定，不保存文件）	
	maxVolumeSizeMB	文件分卷大小（默认不分卷，单位：MB）	
	forceClose	是否强制停止文件保存（默认是 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

9.2.8 数据分析

9.2.8.1 强度分析

各通道实时计数率的 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；
- 暂时仅支持计数率统计窗口默认为 1s。

【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

int getSingleCountRate(int channel)	channel	通道号	获取指定通道的实时计数率

9.2.8.2 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 ALEA
```

```
    ChronosInst alea =    ChronosInst::create("apiconfig.ini", "D:\\cinstapi");

    // 2. 设置错误回调
    alea.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 = alea.getDevices();
    if(!devices.empty()) {
        alea.connect(devices[0].type);
    }

    // 4. 启动工作线程，执行配置和操作
    ...
    // 设置数据模式
    INST_RESULT<int> result = alea.setDataMode(INST_DATA_MODE_SM3);

    // 设置数据打包格式
    bool flag = alea.setDataPackingFormat(INST_DATA_PACKING_BITWISE);

    // 采集原始数据
    flag = alea.acqRawData(INST_DATA_MODE_SM3, 10);

    // 实时读取数据
    std::vector<char> outData;
    flag = alea.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 = alea.stopAcq();

    ...
    alea.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：ALEA 真随机数生成器设备）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（暂不支持）

【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

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
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	
is_channel_enabled	channel：通道号	bool	获取通道使能状态	

9.3.7 全局配置

【数据模式定义】

定义	值	说明
AleaDataMode.ENTROGY_SOURCE	0	熵源模式
AleaDataMode.SM3	2	SM3 模式

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
set_data_mode	data_mode：数据模式	bool	设置数据模式	详见 AleaDataMode 定义
set_board_id	board_id：板卡 ID	bool	设置新板卡 ID	0~255
tdc_resetn	-	bool	TDC 复位	
system_resetn	-	bool	系统复位	
get_data_mode	-	int	获取数据模式	
get_board_id	-	int	获取板卡 ID	

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
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
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

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；
- 支持二种数据打包格式，可以根据实际需要，选择合适的格式类型；按位打包格式，数据最紧凑，数据采集效率最高。

【数据打包格式定义】

定义	值	说明
DataPackingFormat.DATA_PACKING_BITWISE	0	按位打包
DataPackingFormat.DATA_PACKING_BYTEWISE	1	按字节打包

【API 定义】

API 函数	参数	返回值	说明
set_data_packing_format	data_packing_format	bool	设置数据打包格式类型
get_data_packing_format	-	int	获取数据打包格式类型
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；	bool	开始进行数据采集（异步模式） <data_mode> 请详见前面的 AleaDataMode 定义 <duration> 采集时间：单位-秒 <file_name> 未明确后缀名情况下，默认后缀名.bin；

	False-等待内存中数据全部保存）		<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_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

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。

【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	
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	
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, AleaDataMode,
DataPackingFormat
import time
try:
        # ensure config path is absolute or exists; ChronosInst will throw error if not found
        alea = ChronosInst("D:\\cinstapi\\config\\apiconfig.ini", "D:\\test")

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

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

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

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

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

    # process channel config
    res = alea.disable_channel(2)
    print("disable_channel->:", res)
    ......

    # process global config
    res = alea.set_data_mode(AleaDataMode.SM3)
```

```
        print("set_data_mode->:", res)
        ......

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

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

        # acq raw data
        alea.set_data_packing_format(DataPackingFormat.DATA_PACKING_BITWISE)
        res = alea.acq_rawdata(AleaDataMode.SM3, 10)
        print("acq_rawdata->:", res)
        raw_bytes = alea.fetch_rawdata(500)
        time.sleep(10)
        alea.stop_acq()
        time.sleep(2)
        ...

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

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

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

            alea.stop()   # stop and clean up
except RuntimeError as e:
```

```
    print(f"ALEA 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：ALEA 真随机数生成器设备）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（暂不支持）

【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

【API 定义】

API 函数	参数	返回值	功能描述	有效范围
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	
isChannelEnabled	channel：通道号	logical	获取通道使能状态	

9.4.7 全局配置

【数据模式定义】

定义	值	说明
AleaDataMode.ENTROGY_SOURCE	0	熵源模式
AleaDataMode.SM3	1	SM3 模式

【API 定义】

API 函数	参数	返回值	功能描述	有效范围

setDataMode	data_mode: 数据模式	logical	设置数据模式	详见 AleaDataMode 定义
setBoardId	board_id: 板卡 ID	logical	设置新板卡 ID	0~255
tdcResetn	-	logical	TDC 复位	
systemResetn	-	logical	系统复位	
getDataMode	-	int32	获取数据模式	
getBoardId	-	int32	获取板卡 ID	

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
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
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

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；
- 支持二种数据打包格式，可以根据实际需要，选择合适的格式类型；按位打包格式，数据最紧凑，数据采集效率最高。

【数据打包格式定义】

定义	值	说明
DataPackingFormat.DATA_PACKING_BITWISE	0	按位打包

DataPackingFormat.DATA_PACKING_BYTEWISE	1	按字节打包

【API 定义】

API 函数	参数	返回值	说明
setDataPackingFormat	data_packing_foramt: 数据打包格式类型	logical	设置数据打包格式类型
getDataPackingDataFormat	-	int	返回数据打包格式类型
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 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）	logical	开始数据采集 <data_mode> 请详见前面的 AleaDataMode 定义 <duration> 采集时间：单位-秒 <file_name> 未明确后缀名情况下，默认后缀名.bin；
fetchRawData	timeout_ms：超时时间（单位：ms）	(success, rawData])	实时读取 RAW 数据 <rawData> 是 uint8([])数组
isRawDataProcessing	-	logical	是否在处理 RAW 数据
stopAcq	-	logical	停止数据采集

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)	
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	
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(@() alea.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_alea_example.m
% ChronosInst Matlab 使用示例

function chronos_alea_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 ALEA
    fprintf('\n2. Initializing ALEA...\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 ALEA 实例
    alea = ChronosInst(configPath, dllPath);

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

    fprintf('ALEA initialized successfully!\n');
catch ME
    fprintf('Failed to initialize ALEA: %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 = alea.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 = alea.connect(deviceType);
        if success
            fprintf('Connected to device successfully!\n');
            %% 5. 获取板卡 ID
            fprintf('\n5. Getting board ID...\n');
            boardId = alea.getBoardId();
            fprintf('Board ID: %d\n', boardId);

            %% 6. 设备自检操作示例
            fprintf('\n6. Device self-check operations...\n');
            result = alea.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');
            % 通道使能
            result = alea.disableChannel(1);
```

```
result = alea.isChannelEnabled(1);
result = alea.enableChannel(1)
......

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

% 设置数据模式
fprintf('Setting data mode...\n');
result = alea.setDataMode(AleaDataMode.SM3);
if result
        fprintf('Set data mode successfully\n');
else
        fprintf('Set data mode failed\n');
End
......
%% 9. 设备相关数据获取示例
fprintf('\n10. Get device info...\n');

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

%% 10. 网络设置示例
fprintf('\n11. Network configuration control...\n');
result = alea.setIPs("10.0.0.5", "10.0.0.10");
result = alea.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');
% 设置打包格式

alea.setDataPackingFormat(DataPackingFormat.DATA_PACKING_BITWISE);


% 启用采集
fprintf('Starting acquisition...\n');
result = alea.acqRawData(AleaDataMode.SM3, 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 = alea.stopAcq();

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

        % 强度分析
        result = alea.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 = alea.stopSingleCountRateRecording();
        ......

        %% 14. 重置操作示例
        fprintf('\n14. Resetn control...\n');
        result = alea.tdcResetn();
        if result
            fprintf('ALEA resetn successful\n');
        else
            fprintf('Failed to process 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：ALEA 真随机数生成器设备）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（暂不支持）

【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	获取当前可用的设备接口列表	

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

9.5.5 设备自检

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

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciSelfCheck	ChronosInst in: ChronosInst 实例	ChronosInst out: ChronosInst 实例	设备自检	

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

9.5.6 通道配置

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciEnableChannel	ChronosInst in：ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道使能	
ciDisableChannel	ChronosInst in：ChronosInst 实例 channel：通道号 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	设置通道禁能	
ciEnableAllChannels	ChronosInst in：ChronosInst 实例 error in：错误输入	ChronosInst out：ChronosInst 实例 ret：操作状态码 error out：错误输出	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	
ciIsChannelEnabled	ChronosInst in：	ChronosInst	获取通道使	

	ChronosInst 实例 channel：通道号 error in：错误输入	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	能状态	

9.5.7 全局配置

【数据模式定义】

定义	值	说明
DATA_MODE_ENTROGY_SOURCE	0	熵源模式
DATA_MODE_SM3	2	SM3 模式

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
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
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

ciTdcResetn	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	TDC 复位	
ciSystemResetn	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	系统复位	系统复位后，需要等待一段时间再操作
ciGetDataMode	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 data mode: 数据模式类型 error out：错误输出	获取数据模式	
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

9.5.8 状态查询

【设备状态簇】

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

【VI 定义】

VI 名称	Input 输入	Output 输出	功能描述	备注
ciGetChannelNum	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 channel num: 设备通道数 error out：错误输出	获取通道数量	获取设备通道数，不包含参考通道
ciGetDeviceType	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 device type: 设备类型 error out：错误输出	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)
ciGetDeviceMode	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 device mode:	获取设备型号	0x1: Venus Lite-32/Mercury-16/Alea-16 0x2: Venus Lite-24/Mercury-12 0x3: Venus Lite-

		设备型号 error out：错误输出		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：错误输出	获取设备状态	包含温度和各电路电流，详见前面簇定义
ciGetDeviceTemp	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 device temp:	获取设备核心温度	

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

9.5.9 网络配置

注意事项：

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

【VI 定义】

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

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

		输出		

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；
- 支持二种数据打包格式，可以根据实际需要，选择合适的格式类型；按位打包格式，数据最紧凑，数据采集效率最高；
- 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
ciSetDataPackingFormat	ChronosInst in: ChronosInst 实例 data packing format:数据打包格式类型 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	设置数据打包格式类型	0-按位打包 1-按字节打包
ciGetDataPackingFormat	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 data packing foramt：数据打包格式类型 error out：错误输出	获取数据打包格式类型	
ciAcqRAWData	ChronosInst in: ChronosInst 实	ChronosInst out: ChronosInst 实例	开始进行数据采集（异	<data mode>

	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（默认是false-等待内存中数据全部保存） error in：错误输入	ret：操作状态码 error out：错误输出	步模式），数据保存在Server 服务器端	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，默认后缀名.bin； <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	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

	false-等待内存中数据全部保存） error in：错误输入			
ciFetchRAWData	ChronosInst in: ChronosInst 实例 timeout(ms)：超时时间（单位：毫秒） error in：错误输入	ChronosInst out: ChronosInst 实例 data: U8 字节数组 error out：错误输出	实时读取 Raw 数据 返回 U8 字节数组	
ciStopAcq	ChronosInst in: ChronosInst 实例 error in：错误输入	ChronosInst out: ChronosInst 实例 ret：操作状态码 error out：错误输出	停止数据采集	
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（默认为 0） force close：是否强制停止文件保存（默认是	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

	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	
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	
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。

10. 典型测试电路和测试结果

10.1 测试简介

本章主要介绍 ALEA16 系统的测试方案与核心结果。

由于 ALEA 采用了 RO 振荡器和 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。

除温度测试外，所有测试在实验室常温（~25 ℃）环境下进行。

10.2 RO 周期测试

由于 ALEA 采取了 RO 方式作为熵源，每个设备、每个通道的计数率会有不同。如下表是一个设备各个通道的典型测试结果。

表格 7 各通道熵比特率（典型结果）

通道号	计数率，Mbps
1.	112.3
2.	111.7
3.	100.5
4.	107.6
5.	107.2

6.	107.5
7.	120.1
8.	119.5
9.	120.5
10.	109.3
11.	111.4
12.	114.8
13.	111.9
14.	115.9
15.	125.3
Total	1695.5

实际采集数据量统计：

对于熵源数据模式，系统采集 2s 数据，采集 10 次，平均每次可以采集到 435000KB 随机数，即熵源随机数率约为 1.7 Gbps；

对于 SM3 数据模式，系统采集 2s 数据，采集 10 次，平均每次可以采集到 340000KB 随机数，即熵源随机数率约为 1.4 Gbps；

10.3 熵源输出测试

熵源模式，ALEA 输出没有经过任何后处理的随机数据流。

10.3.1 NIST SP 800-90B 测试

NIST SP 800-90B《Recommendation for the Entropy Sources Used for Random Bit Generation》是美国国家标准与技术研究院（NIST）于 2018 年发布的熵源评估标准，与 SP 800-90A（DRBG）和 SP 800-90C（RBG 构造）共同构成随机数生成器的完整规范体系。该标准将熵源划分为噪声源、健康检测和调理三个部分，并要求通过统计估计给出熵源的最小熵下界：若熵源通过 IID 检验，则采用 IID 轨（以 MCV 估计）；否则采用 non-IID 轨，用 MCV、Collision、Markov、Compression、t-Tuple、LRS、MultiMCW、Lag、MultiMMC、LZ78Y 共 10 个估计器分别估计并取最小值。标准同时规定了启动检测与连续检测（RCT、APT）等健康检测机制，以及基于调理定理的熵值折算方法，是当前 TRNG

设计与认证领域最具影响力的评估依据之一。

10.3.1.1 IID 测试

为判断熵源是否满足独立同分布（IID）假设，NIST SP 800-90B 规定了一组统计检验。首先，卡方独立性检验用于检测样本之间是否存在相关性；卡方拟合优度检验用于检测样本分布是否偏离均匀分布；最长重复子串（LRS）检验用于检测数据中是否出现异常长的重复模式；置换检验则通过 19 种统计量（游程、碰撞、周期性、协方差、压缩等），将原始序列与随机置换序列进行对比，以判断其是否含有非随机结构。所有检验均在 1,000,000 个样本上执行，各检验的 p 值须高于标准规定的显著性水平，或通过规定的置换轮数判据。若全部检验通过，则说明熵源满足 IID 假设，可采用 IID 轨、以最常值（MCV）估计最小熵；否则必须转入 non-IID 轨，由 10 个估计器分别估计并取最小值。因此，IID 检验不仅是对熵源统计质量的直接验证，也是决定后续熵估计方法的关键前提。

在室温下，ALEA 设备输出的熵源随机数 IID 测试结果如下表所示。最小熵为 0.994646 bit。

表格 8 IID 测试结果（熵源）

参数	值	说明
样本数	1000000	
Raw Mean	0.499429	原始均值
Median	0.5	中位数
Binary	TRUE	二值数据
MCV mode	500571	最常见值出现次数
MCV p-hat	0.500571	最常见值概率
MCV p_u	0.50185891	99% 上界
MCV H_original	0.994646	IID 轨最小熵
Chi-square independence score	2117.141997	独立性卡方统计量
Chi-square independence dof	2046	自由度
Chi-square independence p-value	0.133522	>0.01 通过
Chi-square goodness score	6.486204	拟合优度统计量

Chi-square goodness dof	9	自由度
Chi-square goodness p-value	0.690442	>0.01 通过
Chi-square tests	Passed	卡方测试结论
LRS P_col	0.500001	最长重复子串 P_col
LRS Length	37	最长重复子串长度
LRS Pr(X>=1)	0.973692	LRS 概率
LRS test	Passed	LRS 结论
Permutation tests	Passed (19/19)	置换检验结论
Overall IID	Passed	总体结论

10.3.1.2 non_IID 测试

当熵源未能通过 IID 检验，或其统计特性不宜作独立同分布假设时，NIST SP 800-90B 要求采用 non-IID 轨进行熵估计。该轨不依赖 IID 假设，而是从多个角度对序列进行最坏情况分析：MCV 估计最常值概率，Collision 估计碰撞概率，Markov 估计一阶马尔可夫转移概率，Compression 估计序列的可压缩性，t-Tuple 与 LRS 分别从短元组和最长重复子串的角度刻画局部重复结构，MultiMCW、Lag、MultiMMC 和 LZ78Y 则通过多种预测器评估序列的可预测性。每个估计器独立给出一个最小熵估计值，标准规定取其中最小值作为熵源的最小熵（H_original），从而保证估计结果在未知依赖结构下仍然保守可靠。全部估计均在 1,000,000 个样本上执行，是 TRNG 认证中最严格、最常用的一条评估路径。

在室温下，ALEA 设备输出的熵源随机数 non_IID 测试结果如下表所示。

表格 9 non_IID 测试结果（熵源）

Item	Pmax	hmin
MCV	0.50185891	0.994646
Collision	3.3120E-39	0.998652
Markov	0.53062834	0.914226

T-Tuple	0.50351513	0.989893
LRS	0.50136046	0.99608
Multi-MCW	0.50133642	0.996149
Lag	0.50156392	0.995495
Multi-MMC	0.50118842	0.996575
LZ78Y	0.50185891	0.994646

10.3.2 NIST SP 800-22 测试

美国国家标准与技术研究院（NIST）发布的 SP 800-22《A Statistical Test Suite for Random and Pseudo random Number Generators for Cryptographic Applications》（Rev. 1a，2010）是随机数统计检验领域使用最广泛的标准之一。该标准从频数、块内频数、游程、最长游程、矩阵秩、离散傅里叶变换（频谱）、非重叠模板匹配、重叠模板匹配、Maurer 通用统计、线性复杂度、序列相关、近似熵、累加和、随机游走及其变体等多个角度，规定了 15 项统计检验，用于判断二进制序列是否存在偏离随机性的统计特征。检验流程为：对每条被测序列计算 p 值，若 $p \geq \alpha$（通常取 0.01）则判定该序列通过；当被测数据包含多条序列时，还需统计通过比例，并对全部 p 值进行均匀性检验（二级 Kolmogorov-Smirnov 检验），只有比例与均匀性同时满足判据，才认为该测试项通过。需要指出的是，统计检验只能发现非随机性，不能证明随机性，因此 SP 800-22 通常与 SP 800-90B、AIS-31 等熵源评估标准配合使用，作为随机数发生器统计质量的常规验证手段。

下表为 SP 800-22 熵源随机数分析典型结果，分析数据个数 1024000。20 组 × 1,024,000 比特的 +20°C 熵源数据 15 项测试全部通过。

表格 10 SP800-22 测试结果（熵源）

测试项	P-VALUE（均匀性）	通过比例	判定
Frequency	0.534146	20/20	Pass
BlockFrequency	0.213309	20/20	Pass
CumulativeSums (1)	0.090936	20/20	Pass
CumulativeSums (2)	0.637119	20/20	Pass
Runs	0.275709	20/20	Pass
LongestRun	0.090936	20/20	Pass
Rank	0.991468	18/20	Pass
FFT	0.066882	20/20	Pass
NonOverlappingTemplate (148)	0.017912~0.991468	114×20/20、30×19/20、4×18/20	Pass
OverlappingTemplate	0.534146	20/20	Pass
Universal	0.066882	20/20	Pass
ApproximateEntropy	0.911413	20/20	Pass
RandomExcursions (8)	0.000648~0.275709	17/17	Pass
RandomExcursionsVariant (18)	0.000060~0.275709	17/17	Pass
Serial (2)	0.534146 / 0.834308	20/20	Pass
LinearComplexity	0.437274	20/20	Pass

10.3.3 Dieharder 测试

Dieharder 是由美国杜克大学 Robert G. Brown 等人开发的开源随机数统计测试套件，在 Marsaglia 的 Diehard 测试集基础上扩展而来，包含经典 Diehard 17 项、RGB 系列以及 STS 系列等 30 余项测试，是目前随机数统计检验中使用最广泛的工具之一。测试时，将待测数据以原始二进制形式（1 字节 8 比特）输入，由 dieharder 逐项读取并计算 p 值：若 $p \geq 0.01$，则该测试项判定为 PASSED（通过），否则判定为 FAILED（失败）；当同一测试重复多次时，还会对所得 p 值的均匀性进行进一步检验。本文对 ALEA16 熵源输出数

据执行了 dieharder 全部测试，受篇幅限制，正文仅列出经典 Diehard 17 项（test 0~16）的测试结果，RGB 系列、STS 系列等其余测试项的结果从略。

表格 11 Dieharder 测试结果（熵源）

编号	测试名称	p-value	评估
0	diehard_birthdays	0.6889669	PASSED
1	diehard_operm5	0.10689792	PASSED
2	diehard_rank_32x32	0.24253882	PASSED
3	diehard_rank_6x8	0.93919696	PASSED
4	diehard_bitstream	0.91781414	PASSED
5	diehard_opso	0.75276858	PASSED
6	diehard_oqso	0.87287703	PASSED
7	diehard_dna	0.32001859	PASSED
8	diehard_count_1s_str	0.54541444	PASSED
9	diehard_count_1s_byte	0.69708194	PASSED
10	diehard_parking_lot	0.91461926	PASSED
11	diehard_2dsphere	0.97526702	PASSED
12	diehard_3dsphere	0.87881951	PASSED
13	diehard_squeeze	0.02563109	PASSED
14	diehard_sums	0.3200126	PASSED
15	diehard_runs	0.64010050 / 0.34986103	PASSED
16	diehard_craps	0.00837601 / 0.08745911	PASSED

10.3.4 AIS-31 测试

德国联邦信息安全局（BSI）发布的 AIS-31《Functionality Classes for Random Number Generators》是另一套被广泛采用的随机数评估标准，与 NIST SP 800-90B 互为补充。AIS-31 将随机数发生器划分为 PTG.2、PTG.3（物理真随机）和 DRG（确定性）等功能类，并规定了一套统计检验流程：T0 不相交性检验、T1 单比特检验、T2 扑克检验、T3 游程检验、T4 长游程检验、T5 自

相关检验、T6 均匀分布检验、T7 多项分布检验和 T8 熵检验。其中 T1T8 在更大样本上执行，用于评估分布均匀性与熵水平。所有检验须同时满足标准规定的上下界，才能判定该熵源通过 AIS-31 评估。

下表为 AIS-31 熵源随机数分析典型结果，分析数据个数 1E6。所有测试均通过。

表格 12 AIS-31 测试项和结果（熵源）

AIS-31	Test Result
Disjointness Test (TO)	PASS
Monobit Tests (T1)	PASS
Poker Tests (T2)	PASS
Run Tests (T3)	PASS
Long Run Test (T4)	PASS
Autocorrelation Test (T5)	PASS
Uniform Distribution Test (T6)	PASS
Multinomial Distributions (T7)	PASS
Entropy Test (T8)	PASS

10.3.5 分布均一性测试

理想情况下，TRNG 产生 0 和 1 的概率应当完全相同。序列偏置会削弱随机性，并使 TRNG 易受攻击。我们用下式评估随机序列的偏置大小：

$$Bias = \frac{|N_1 - N_2|}{N_1 + N_2} \times 100\%$$

其中 N1 和 N2 分别为比特 1 和比特 0 的个数。实测的统计偏差为 0.029%。

随机比特流还被转换为二维比特流图像，如下图所示。在左图中，比特 1 渲染为黑色像素，而在右图中则用白色像素表示。黑白像素分布均匀，验证了所生成的随机数具有均匀性和低偏置。

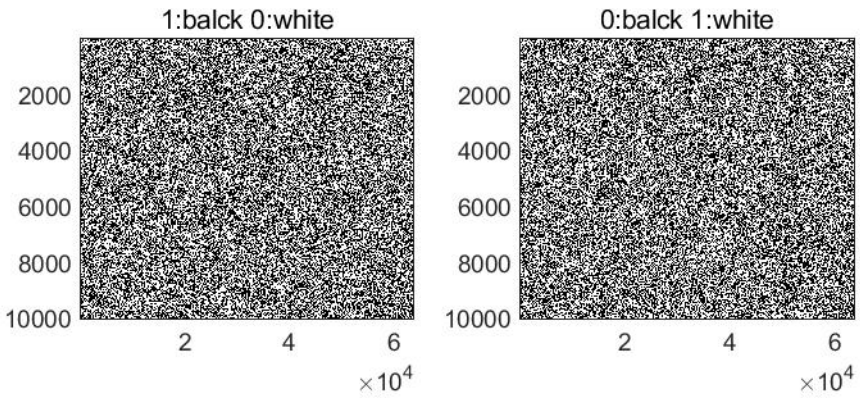


图 24 比特 0 和 1 二维分布图 9（熵源）

10.3.6 温度漂移测试

10.3.6.1 测试平台

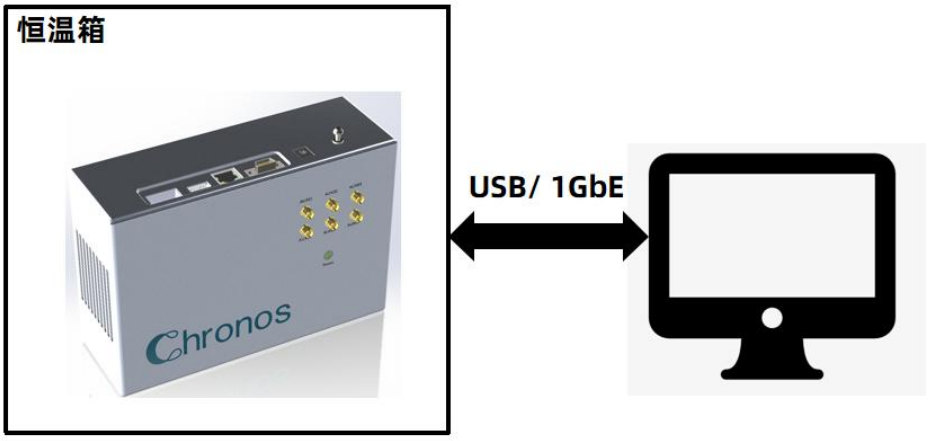


图 25 温漂测试平台示意图

为评估温度对随机数发生器性能的影响，搭建了如图所示的温漂测试平台。平台由恒温箱、ALEA 随机数发生器样机和上位机三部分组成：样机置于恒温箱内，通过 USB 3.0 或 1 GbE 接口与箱外上位机连接，实现随机数数据的实时采集与存储。测试时，将恒温箱分别设定为 −10 °C、0 °C、+10 °C、+20 °C、+30 °C 和 +40 °C 六个温度点，待箱内温度稳定并使样机充分热平衡后，由上位机连续采集原始熵源比特流与 SM3 调理后的输出比特流，每组数据保存为独立的二进

制文件；随后使用 NIST SP 800-90B 参考工具（ea_iid 与 ea_non_iid）对两组数据分别进行 IID 与非 IID 分析，得到各温度点下的最小熵估计值。样机的 FPGA 内核温度可通过片上 XADC 实时读取，用于记录环境温度与芯片结温的对应关系。

10.3.6.2 IID 测试

NIST SP 800-90B IID 测试评估随机性温漂。测试结果如下表所示，每次测试分析数据个数 1E6。

表格 13 IID 温漂测试结果（熵源）

环境温度(℃)	FPGA 核心温度(℃)	Raw Mean	MCV p_u	H_original	Chi2 ind p	Chi2 GOF p
-10	30	0.499823	0.50146492	0.995779	0.621915	0.490764
0	38	0.498922	0.50236591	0.993190	0.082079	0.752949
+10	46	0.500869	0.50215691	0.993790	0.229684	0.802009
+20	55	0.499429	0.50185891	0.994646	0.133522	0.690442
+30	60	0.499959	0.50132892	0.996171	0.708690	0.254620
+40	68	0.500166	0.50145392	0.995811	0.026492	0.189058

最小熵与外界环境温度的关系如下图所示。

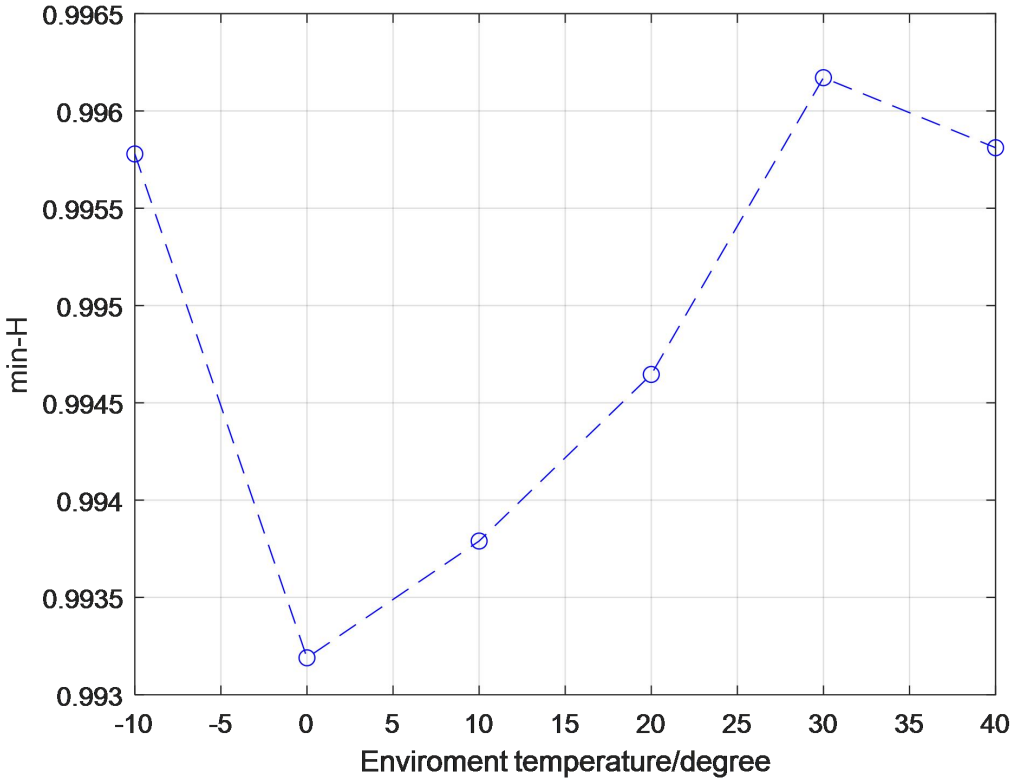


图 26 IID 最小熵与环境温度测试结果（熵源）

10.3.6.2 non_IID 测试

NIST SP 800-90B non_IID 熵源测试评估随机性温漂。测试结果如下表所示，每次测试分析数据个数 1E6。

表格 14 NIST SP 800-90B 温漂测试结果（熵源）

温度(°C)	FPGA 核心温度(°C)	MCV	Collision	Markov	T-Tuple	LRS	Multi-MCW	Lag	Multi-MMC	LZ78Y
-10	30	0.995779	0.934754	0.997759	0.94154	0.976287	0.99692	0.995646	0.997167	0.995436
0	38	0.99319	0.917854	0.997036	0.919089	0.994334	0.99635	0.994777	0.995161	0.993694
10	46	0.99379	0.947521	0.997728	0.921623	0.994988	0.997767	0.998052	0.995555	0.996981
20	55	0.994646	0.914499	0.998652	0.914226	0.989893	0.99608	0.996149	0.995495	0.996575
30	60	0.996171	0.913806	0.99884	0.921623	0.977339	0.996514	0.996699	0.994396	0.99773
40	68	0.995811	0.944247	0.99896	0.93978	0.99063	0.994762	0.995507	0.995345	0.996345

10.3.7 重启测试

ALEA 设备经过 1000 次上电重启，采集超过 1E6 个随机数序列，测试 NIST SP 800-90B IID，其最小熵均大于 0.99 bit。

10.4 SM3 输出测试

10.4.1 NIST SP 800-90B 测试

10.4.1.1 IID 测试

在室温下，ALEA 设备输出的随机数 IID 测试结果如下表所示。最小熵为 0.996243 bit。

表格 15 IID 测试结果

参数	值	说明
样本数	1000000	
Raw Mean	0.500016	原始均值
Median	0.5	中位数
Binary	TRUE	二值数据
MCV mode	500016	最常见值出现次数
MCV p-hat	0.500016	最常见值概率
MCV p_u	0.50130392	99% 上界
MCV H_original	0.996243	IID 轨最小熵
Chi-square independence score	1984.019318	独立性卡方统计量
Chi-square independence dof	2046	自由度
Chi-square independence p-value	0.833586	>0.01 通过
Chi-square goodness score	8.780496	拟合优度统计量
Chi-square goodness dof	9	自由度
Chi-square goodness p-value	0.457778	>0.01 通过
Chi-square tests	0.5	卡方测试结论
LRS P_col	38	最长重复子串 P_col
LRS Length	0.837788	最长重复子串长度
LRS Pr(X>=1)	19/19	LRS 概率
LRS test	Passed	LRS 结论
Permutation tests	1000000	置换检验结论
Overall IID	0.500016	总体结论

10.4.1.2 non_IID 测试

下表为 NIST SP 800-90B non_IID 随机数分析典型结果，分析数据个数 1E6。

表格 16 NIST SP 800-90B 测试结果

Item	Pmax	hmin
MCV	0.50130392	0.996243
Collision	3.4751E-39	0.99811
Markov	0.52884286	0.919089
T-Tuple	0.50248909	0.992836
LRS	0.50092743	0.997326
Multi-MCW	0.50105642	0.996955
Lag	0.50096392	0.997221
Multi-MMC	0.50155443	0.995522
LZ78Y	0.50130392	0.996243

10.4.2 NIST SP 800-22 测试

下表为 SP 800-22 随机数分析典型结果，分析数据个数 1024000。20 组 × 1,024,000 比特的 +20℃ 熵源数据 15 项测试全部通过。

表格 17 SP800-22 测试结果

测试项	P-VALUE（均匀性）	通过比例	判定
Frequency	0.437274	20/20	Pass
BlockFrequency	0.637119	20/20	Pass
CumulativeSums (1)	0.162606	20/20	Pass
CumulativeSums (2)	0.162606	20/20	Pass
Runs	0.122325	20/20	Pass
LongestRun	0.911413	20/20	Pass
Rank	0.048716	20/20	Pass
FFT	0.911413	20/20	Pass
NonOverlappingTemplate (148)	0.006196~1.000000	18/20×4; 19/20×19; 20/20×125	Pass
OverlappingTemplate	0.437274	20/20	Pass
Universal	0.035174	20/20	Pass

ApproximateEntropy	0.534146	20/20	Pass
RandomExcursions (8)	0.001399~0.275709	17/17	Pass
RandomExcursionsVariant (18)	0.000005~0.437274	16/17×1; 17/17×17	Pass
Serial (2)	0.739918	20/20	Pass
LinearComplexity	0.834308	20/20	Pass

10.4.3 Dieharder 测试

本文对 ALEA16 熵源输出数据执行了 dieharder 全部测试，受篇幅限制，正文仅列出经典 Diehard 17 项（test 0~16）的测试结果，RGB 系列、STS 系列等其余测试项的结果从略。

表格 18 Dieharder 测试结果

编号	测试名称	p-value	评估
0	diehard_birthdays	0.54571716	PASSED
1	diehard_operm5	0.01692343	PASSED
2	diehard_rank_32x32	0.8719463	PASSED
3	diehard_rank_6x8	0.72873507	PASSED
4	diehard_bitstream	0.95568182	PASSED
5	diehard_opso	0.20530157	PASSED
6	diehard_oqso	0.61183805	PASSED
7	diehard_dna	0.07019509	PASSED
8	diehard_count_1s_str	0.50886925	PASSED
9	diehard_count_1s_byte	0.00324662	WEAK
10	diehard_parking_lot	0.77995133	PASSED
11	diehard_2dsphere	0.9225098	PASSED
12	diehard_3dsphere	0.23210984	PASSED
13	diehard_squeeze	0.15506921	PASSED
14	diehard_sums	0.91045352	PASSED
15	diehard_runs	0.15470826 / 0.33694933	PASSED
16	diehard_craps	0.04814346 / 0.00127811	PASSED

10.4.4 AIS-31 测试

下表为 AIS-31 熵源随机数分析典型结果，分析数据个数 1E6。所有测试均通过。

表格 19 AIS-31 测试项和结果

AIS-31	Test Result
Disjointness Test (TO)	PASS
Monobit Tests (T1)	PASS
Poker Tests (T2)	PASS
Run Tests (T3)	PASS
Long Run Test (T4)	PASS
Autocorrelation Test (T5)	PASS
Uniform Distribution Test (T6)	PASS
Multinomial Distributions (T7)	PASS
Entropy Test (T8)	PASS

10.4.5 分布均一性测试

随机比特流还被转换为二维比特流图像，如下图所示。在左图中，比特 1 渲染为黑色像素，而在右图中则用白色像素表示。黑白像素分布均匀，验证了所生成的随机数具有均匀性和低偏置（~0.004%）。

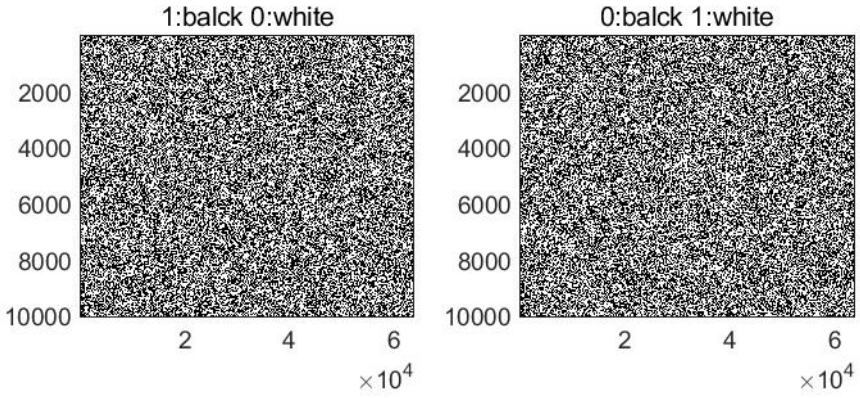


图 27 **比特 0 和 1 二维分布图**

10.4.6 温度漂移测试

10.4.6.1 IID 测试

NIST SP 800-90B IID 测试评估随机性温漂。测试结果如下表所示，每次测试分析数据个数 1E6。

表格 20 IID 温漂测试结果

环境温度(℃)	FPGA 核心温度(℃)	Raw Mean	MCV p_u	H_original	Chi2 ind p	Chi2 GOF p
-10	30	0.499912	0.50137592	0.996035	0.519221	0.611226
0	38	0.500567	0.50185491	0.994658	0.158117	0.96485
+10	46	0.499838	0.50144992	0.995822	0.307809	0.618412
+20	55	0.500016	0.50130392	0.996243	0.833586	0.457778
+30	60	0.49992	0.50136792	0.996058	0.556855	0.671757
+40	68	0.500027	0.50131492	0.996211	0.414055	0.216959

最小熵与外界环境温度的关系如下图所示。

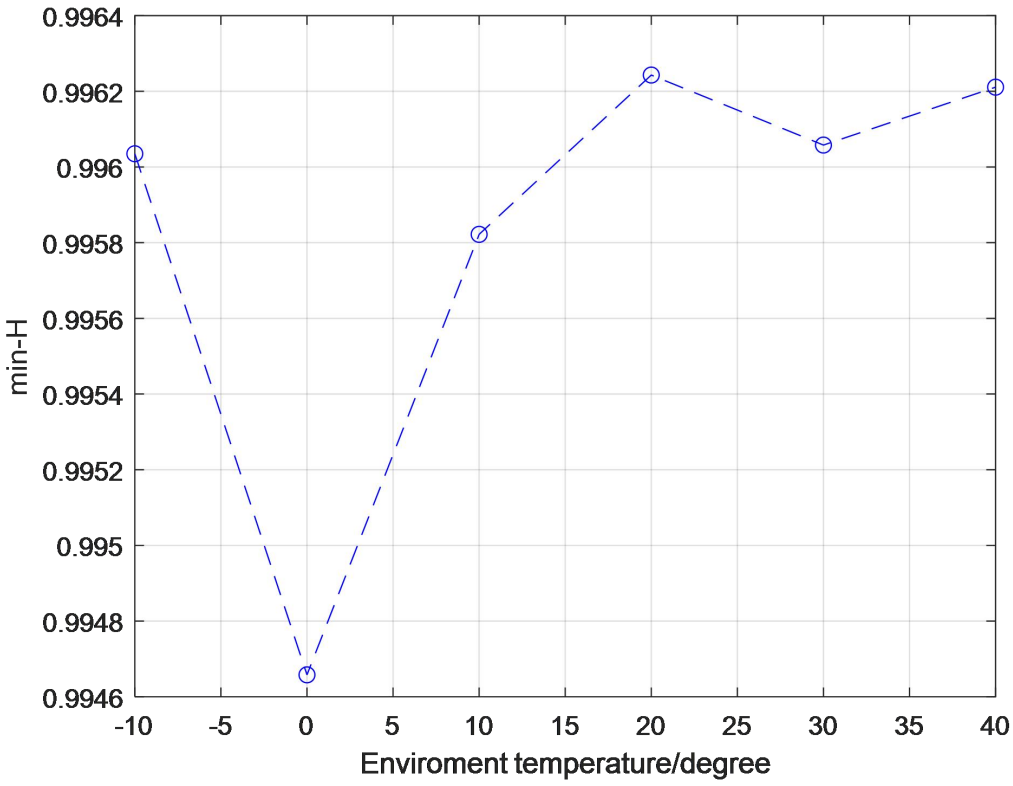


图 28 IID 最小熵与环境温度测试结果

10.4.6.1 non_IID 测试

NIST SP 800-90B non_IID 测试评估随机性温漂。测试结果如下表所示，每次测试分析数据个数 1E6。

表格 21 NIST SP 800-90B 温漂测试结果

温度(°C)	FPGA 核心温度(°C)	MCV	Collision	Markov	T-Tuple	LRS	Multi-MCW	Lag	Multi-MMC	LZ78Y
-10	30	0.996035	0.926452	0.999769	0.93978	0.970832	0.996912	0.996592	0.99615	0.996915
0	38	0.994658	0.88959	0.99634	0.921623	0.994165	0.995775	0.996267	0.995023	0.996011
10	46	0.995822	0.891818	0.998117	0.921623	0.987786	0.99633	0.998883	0.996081	0.996995
20	55	0.996243	0.90184	0.99811	0.919089	0.992836	0.997326	0.996955	0.997221	0.995522
30	60	0.996058	0.914083	0.998505	0.936354	0.996326	0.995959	0.996995	0.997339	0.99894
40	68	0.996211	0.879668	0.997617	0.914226	0.996096	0.995712	0.998444	0.994939	0.995217

10.4.7 重启测试

ALEA 设备经过 1000 次上电重启，采集超过 1E6 个随机数序列，分析其 NIST SP 800-90B IID 测试，其最小熵均大于 0.99 bit。

11. 测量类型原始数据结构

ALEA 分为 2 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。

ALEA 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 22 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）					

其中 Raw data 为真随机数比特流。

表格 23 随机数据格式说明

随机数 [63:0]

12. 支持定制化和二次开发

ALEA 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。

13. 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。

表格 24 产品校准和维护建议

序号	参数	可能出现异常	建议校准周期	官方维护方式
1.	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
2.	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
3.	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
4.	SMA 连接器磨损老化	信号插损加大等	12 个月	信号幅度和前沿测试+连接器更换
5.	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

14. 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。

附录

A. 官方提供设备和附件列表

表格 25 ALEA 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.	ALEA 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	
8.	API 接口函数包	1	基础	官网下载
9.	万兆网卡+光纤转接器	1	选配	需额外付费
10.	定制化需求	1	选配	需额外付费

B. 固件版本说明

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

C. 硬件和软件版本说明

下表描述了 ALEA 硬件和软件版本迭代/进化说明。

表格 26 硬件版本进化说明

硬件版本	发布时间	主要改动	说明
V1.1	2026.10	工程样机	

表格 27 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	2026.11	初版	

D. 对外数据接口列表

表格 28 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. RCT 和 APT 计算公式

RCT 公式如下所示：

$$C = 1 + \left[\frac{-log_2\alpha}{H}\right]$$

- C = 允许的最大连续重复次数；
- 连续相同值达到 C 次 → 报警；

APT 公式如下所示：

$$W = 1 + \left[\frac{-log_2\alpha}{H}\right]$$

- W = 窗口大小；
- 窗口内某个值出现次数 ≥ 阈值 → 报警；
- 二值源（1 比特）的阈值按二项分布算；

本文档历史记录

文档编号	修改时间	修改人	说明
UG-03-1-010-1	2026.11	Cong	V1.0