



CV184X ISP 图像质量调试工具使用手册

Version: 0.1.0

Release date: 2025/05

©2025 北京晶视智能科技有限公司
本文件所含信息归北京晶视智能科技有限公司所有。
未经授权，严禁全部或部分复制或披露该等信息。

目录

1	声明	2
2	概述	3
2.1	工具概述	3
2.2	环境准备说明	4
2.2.1	软硬件需求	4
2.2.2	物理链路连接	4
2.2.3	EVB Uart 连接说明	4
2.2.4	发布包目录说明	4
2.2.5	Linux 系统下板端软件的安装与运行	5
2.3	PC 端软件的安装	6
2.4	快速入门	7
2.4.1	连接单板界面	7
2.4.2	工具主界面	8
2.4.3	常用操作	9
2.4.3.1	保存调试数据文件	9
2.4.3.2	打开调试数据文件	9
2.4.3.3	连接单板	10
2.4.3.4	断开与单板的连接	10
2.4.3.5	图像预览	10
2.4.3.6	抓拍工具	11
2.4.3.7	Bin 导入导出	12
2.4.3.8	标定工具	12
2.4.3.9	辅助工具	13
2.4.3.10	快捷键	13
3	界面及功能说明	14
3.1	在线调试界面及功能说明	14
3.1.1	调试结构界面	14
3.1.2	寄存器/算法参数调整	16
3.1.2.1	寄存器/算法参数的查看与修改	17
3.1.2.2	板端读取数据	18
3.1.2.3	将参数写入到板端	18
3.1.2.4	参数的暂存与恢复	18
3.1.3	调试数据的批量读写与自动写入	19
3.1.4	曲线可视化调试	20
3.1.4.1	通用曲线功能说明	20
3.1.4.2	Gamma 可视化调试	21
3.1.4.3	VO Gamma	22
3.1.5	曝光数据状态和参数设置	23
3.1.5.1	曝光数据状态	23

3.1.5.2	曝光参数设置	24
3.1.6	白平衡数据状态和参数设置	25
3.1.6.1	白平衡数据状态	25
3.1.6.2	白平衡参数设置	25
3.1.7	WDR 曝光参数设置和 Local-tone 开关	26
3.1.7.1	WDR 曝光参数设置	26
3.1.7.2	Local-tone 开关	26
3.1.8	3DNR 和 YNR 降噪调试	27
3.1.9	Y Sharpen 锐化和边缘增强调试	28
3.1.10	Top 页面	29
3.1.11	VPSS Adjustment 页面	29
3.1.12	LDC 页面	30
3.2	ISP 标定工具	30
3.3	高级功能	30
3.3.1	参数的导出导入	30
3.3.2	通讯日志	30
3.4	其他辅助工具使用说明	31
3.4.1	抓拍工具使用说明	31
3.4.1.1	工具界面	31
3.4.1.2	抓取 YUV 图像数据	32
3.4.1.3	抓取 Raw 图像数据	33
3.4.1.4	Dump All	33
3.4.1.5	Caputure Timing	33
3.4.1.6	YUV 图像和 Raw 图像文件名格式说明	34
3.4.1.7	Raw Info 文件格式和内容说明	34
3.4.2	视频播放工具使用说明 (VLC)	35
3.4.2.1	使用 VLC 播放板端视频流步骤	35
3.4.3	辅助对焦工具使用说明	37
3.4.3.1	工具界面	37
3.4.3.2	利用辅助对焦工具进行调焦	37
3.4.3.3	对焦辅助操作	38
3.4.4	3A 分析工具	38
3.4.4.1	工具界面	38
3.4.4.2	获取图像与统计数据	39
3.4.4.3	查看 AE 统计数据	40
3.4.4.4	查看 AWB 统计数据	41
3.4.5	包围曝光	42
3.4.6	连续 raw 工具	43
3.5	参数详细说明	44
4	工具应用参考	50
4.1	工具的参数如何导入导出?	50
4.1.1	使用工具导入导出参数	50
4.1.2	使用库导入导出图像质量参数	51
4.1.3	API 参考	52
4.1.3.1	CVI_BIN_GetBinTotalLen	52
4.1.3.2	CVI_BIN_ExportBinData	53
4.1.3.3	CVI_BIN_ImportBinData	55
4.1.3.4	CVI_BIN_GetSingleISPBinLen	57
4.1.3.5	CVI_BIN_ExportSingleISPBinData	58

4.1.3.6	CVI_BIN_SaveParamToBin	60
4.1.3.7	CVI_BIN_LoadParamFromBin	61
4.1.3.8	CVI_BIN_LoadParamFromBinEx	63
4.1.3.9	CVI_BIN_SetBinName	63
4.1.3.10	CVI_BIN_GetBinName	64
4.1.3.11	CVI_BIN_GetBinExtraAttr	65
4.1.3.12	CVI_ISP_BIN_SetBypassParams	66
4.1.3.13	CVI_ISP_BIN_GetBypassParams	67
4.1.3.14	错误码 1	68
4.1.3.15	错误码 2	68
5	FAQ	69
5.1	为什么工具打开调试数据文件时会提示版本不匹配?	69
5.2	为什么工具连接版子会提示 SoC 不匹配?	70

修订记录

Revision	Date	Description
0.1.0	2025/04/25	Initial

1 声明



法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

联系我们

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 <https://www.sophgo.com/>

技术论坛 <https://developer.sophgo.com/forum/index.html>

2 概述

2.1 工具概述

CviPQ Tool 是专业的图像质量调试工具，将工具跟单板连接以后，提供用户在线调试 ISP 各模块的参数调节，同时还能实时观看参数设置完后的效果。另外，还提供 ISP 标定功能，对需要标定的模块产生各类数据，提供给用户调节参数，获得更佳图像质量。

CviPQ Tool 工具架构如图 2.1 所示，主要分为 PC 端的标定工具在线调试工具 (Tuning Tools)、(Calibration Tools) 和分析工具 (Analysis Tools)，以及抓拍工具 (Dump Tools)。

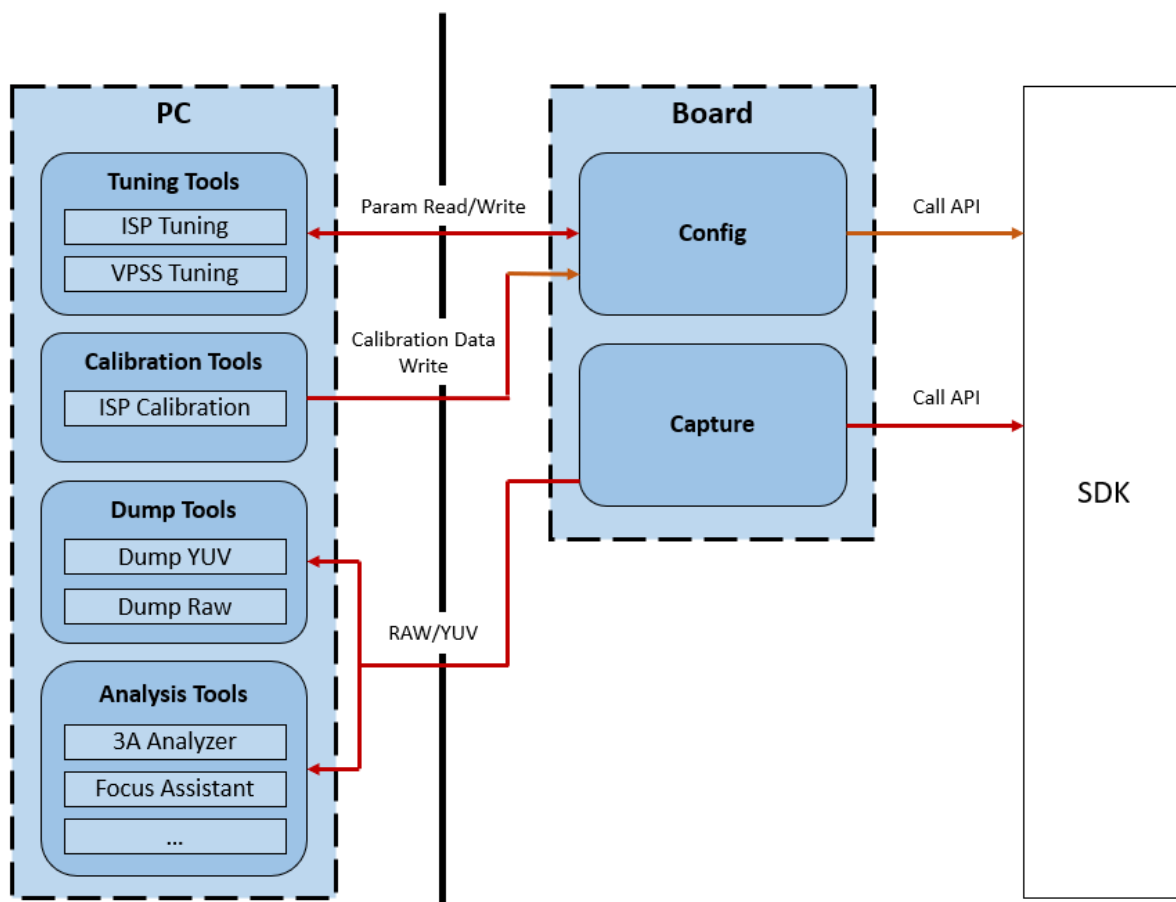


图 2.1: CviPQ Tool 工具架构图

2.2 环境准备说明

2.2.1 软硬件需求

- 硬件需求
 - 台式计算机或便携式计算机
 - 单板硬件（具有网络端口）
 - 网络连接线
 - 显示器的分辨率的高和宽分别至少 1024 和 768
- 软件需求
 - 安装至少 Windows 7 64-bit 或以上的版本
 - 媒体播放器，如 VLC media player

2.2.2 物理链路连接

CviPQ Tool 工具分为 PC 客户端软件和板端服务软件两部分，二者通过网络通信进行交互。

物理链路连接可使用直连和局域网络连接两种方式：

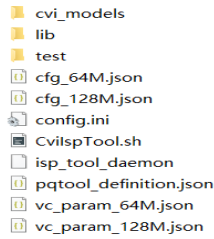
- 直连方式：
 - 使用网线两端分别接入 PC 和板端的网络端口。
- 局域网络方式：
 - 使用网线将板端网络端口接到路由器的本地端口（LAN 口）。
 - 如果 PC 使用无线网络，则按照无线热点的接入方法将 PC 接入到当前路由器的无线热点；如果使用有线网络，则同样地使用网线连接 PC 网络端口和路由器的本地端口（LAN 口）。

2.2.3 EVB Uart 连接说明

请参考“EVB 使用手册”

2.2.4 发布包目录说明

发布包中的 `isp_tool_daemon.tar.gz` 拷贝到板端，解压生成 `install` 目录，目录结构如下图所示。



```
├── cvi_models
├── lib
├── test
├── cfg_64M.json
├── cfg_128M.json
├── config.ini
├── CviIspTool.sh
├── isp_tool_daemon
├── pqtool_definition.json
├── vc_param_64M.json
└── vc_param_128M.json
```

- config.ini 是运行 isp_tool_daemon 时的配置文件，包括配置 log 打印等级、设置 PQbin 默认生成路径、sensor 类型的文件路径。
- CviIspTool.sh 是快速启动 isp_tool_daemon 的脚本文件。

2.2.5 Linux 系统下板端软件的安装与运行

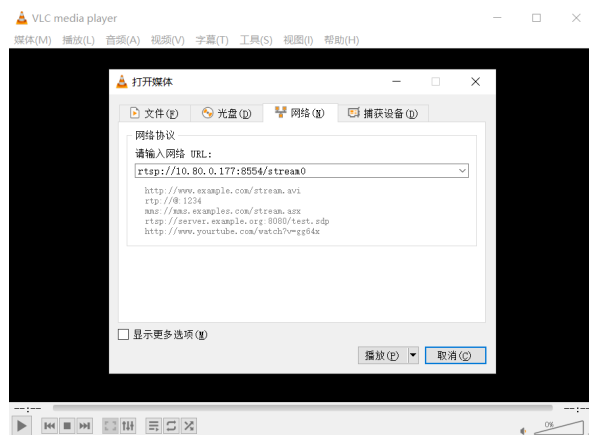
步骤 1. isp_tool_daemon.tar.gz 解压后的目录结构如2.2.4. 发布包目录说明 章节图所示。

步骤 2. 配置板端 IP, ifconfig eth0 xxx.xxx.xxx.xxx

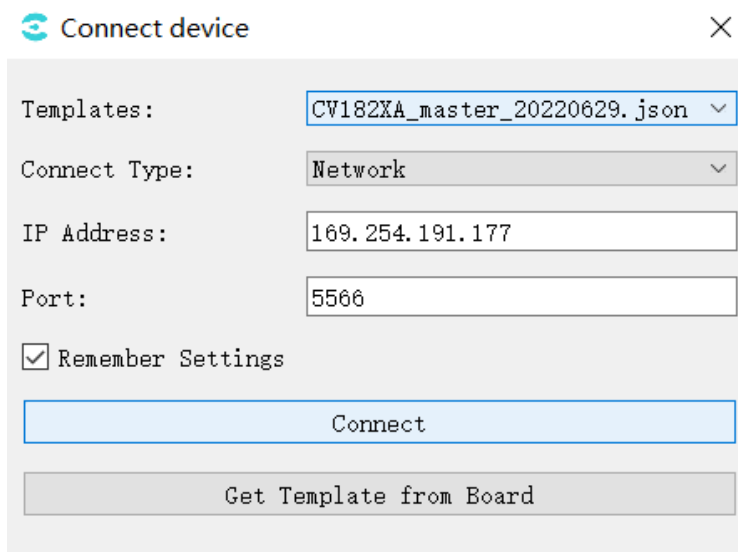
步骤 3. 进入解压后生成 install 目录, cd /mnt/sd/install

步骤 4. 执行 “./CviISPTool.sh” 指令启动板端程序

步骤 5. 连接 VLC 显示影像



步骤 6. 在 PC 端开启 CviPQTool 并且输入板端 IP 即可连上板端



2.3 PC 端软件的安装

CviPQ Tool PC 端软件是绿色免安装软件，只需将软件压缩包解压到任意可写目录，在解压目录下找到 CviPQTool.exe 双击即可运行。

解压后包的目录结构如图 所示。

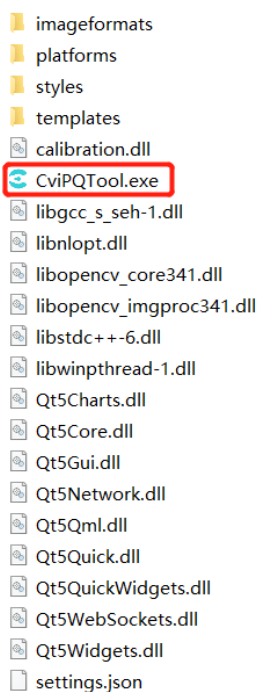


图 2.2: 解压后的目录结构

- CviPQTool.exe 是 CviPQ Tool 的执行程序，直接双击或右键打开即可运行

- settings.json 是 CviPQ Tool 的配置文件，当用户在连接单板接口点选记住设置时参数会被保存到这里
- imageformats/platforms/styles 目录下是 CviPQ Tool 所依赖插件的库文件，其它的 DLL 库文件主要是 Qt 和 opencv 的运行时库

2.4 快速入门

2.4.1 连接单板界面

用户在每次运行 CviPQTool.exe 的时候，会弹出连接单板窗口，如图 2.3 所示，让用户能快速连接单板进行图像质量调试。

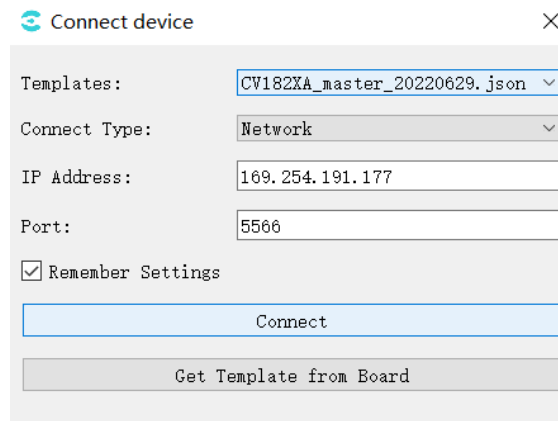


图 2.3: 连接单板接口图例

Templates: 选择 tool ui 模板；

Connect Type: 网络连接；

IP Address: 板端 IP 地址；

Port: 端口号；

Remember Settings: 保存当前窗口设置到 settings.json，下次开启工具自动导入设置。

Get Template from Board: 从板端获取 ui 模板（也可手动将 ui json 文件放到 templates 目录，重启 tool，再从 Templates 下拉框选择）

设置好选项，点击 **Connect** 按钮，进入工具主界面，工具会自动通过网络连接 PC 与单板，并且还会自动从板端读取所有调试项的参数数值。

注解： 注意：开启 tool 时，Templates 中是初始默认 ui 模板，可以点击 Get Template from Board 按钮从板端获取最新匹配的 ui 模板，然后从 Templates 下拉框选择刚获取的 json，再点击 Connect 按钮，等待初始化完 Ui，进入 tool 主界面。

2.4.2 工具主界面

工具主界面如图 所示。

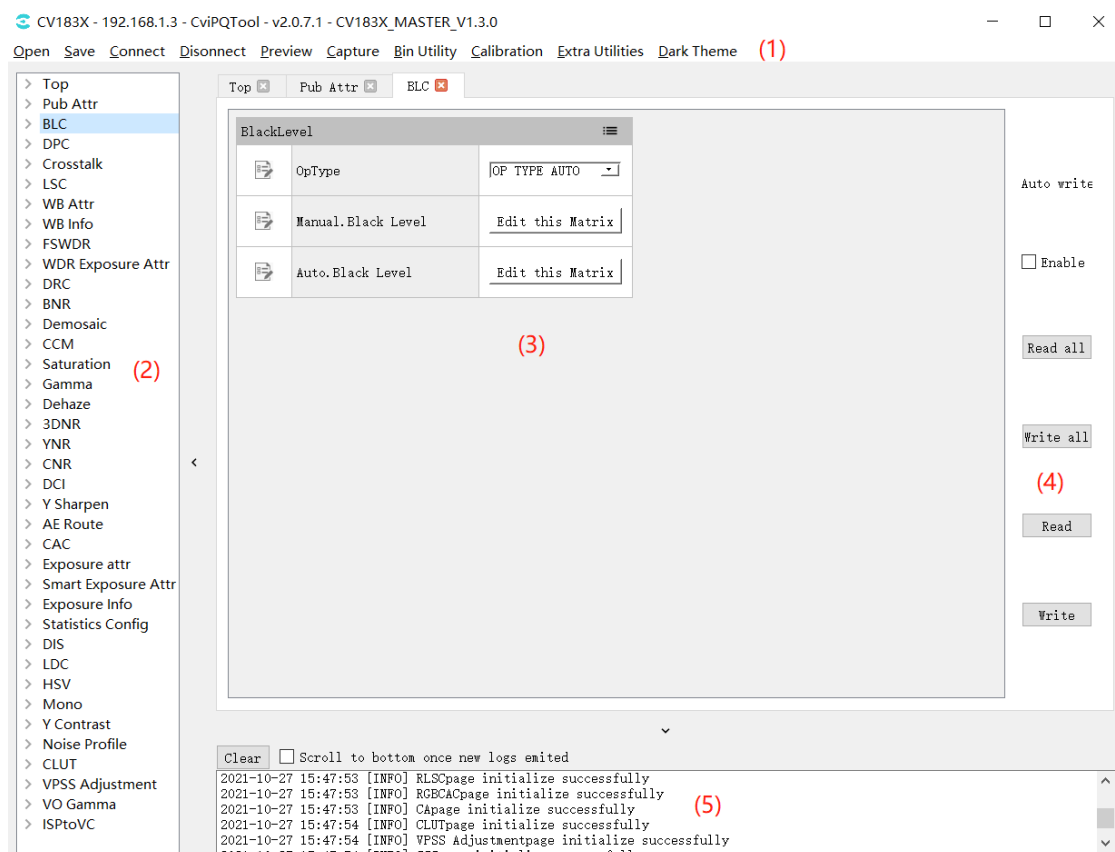


图 2.4: 工具主界面

CviPQ Tool 工具的主界面可以分为以下几个区域:

- (1). 工具栏: 提供一些常用的操作快捷选项
- (2). 调试表面板: 显示所有模块的可调试项
- (3). 调试区域: 此区域会显示由调试表面板中所选中模块对应的调试页面
- (4). 读/写: 可选择自动读写、读写所有页面或当前页面数据
- (5). 提示栏: 显示通信日志

2.4.3 常用操作

2.4.3.1 保存调试数据文件

在工具栏中点击“Save”按钮，会弹出一个选择路径的对话框。当用户选定好一个储存路径时，工具会将当前调试面板表的参数进行保存，保存的文件格式为 *.json，储存参数以及调试表的结构。

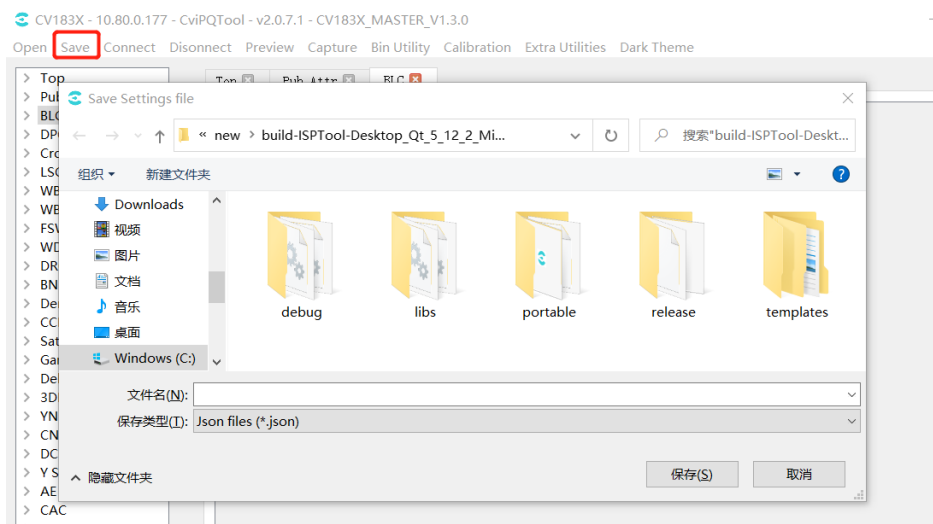


图 2.5: 储存调试数据文件界面图例

2.4.3.2 打开调试数据文件

在工具栏中点击“Open”按钮，会弹出一个对话框，让用户选择要打开的数据文件。当打开数据文件时，工具会将文件中的相关参数加载并将其显示在工具调试面板。

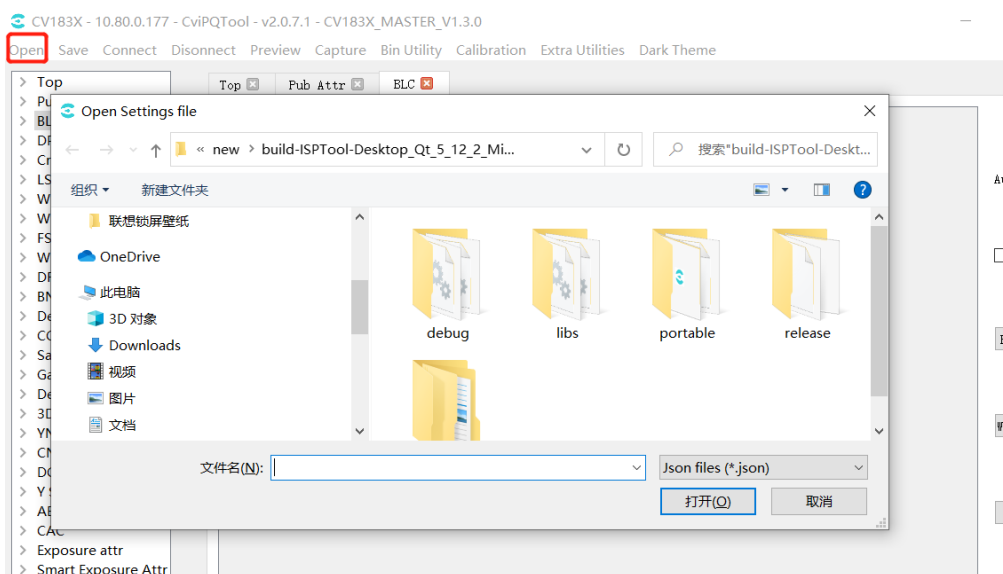


图 2.6: 打开调试数据文件界面图例

2.4.3.3 连接单板

在工具栏中点击“Connect”按钮，会弹出连接单板窗口如图 2.7 所示。用户可在“IP Address”字段中输入单板的 IP 地址，并在“Port”字段中输入埠号后，点击“Connect”，使工具连接单板。

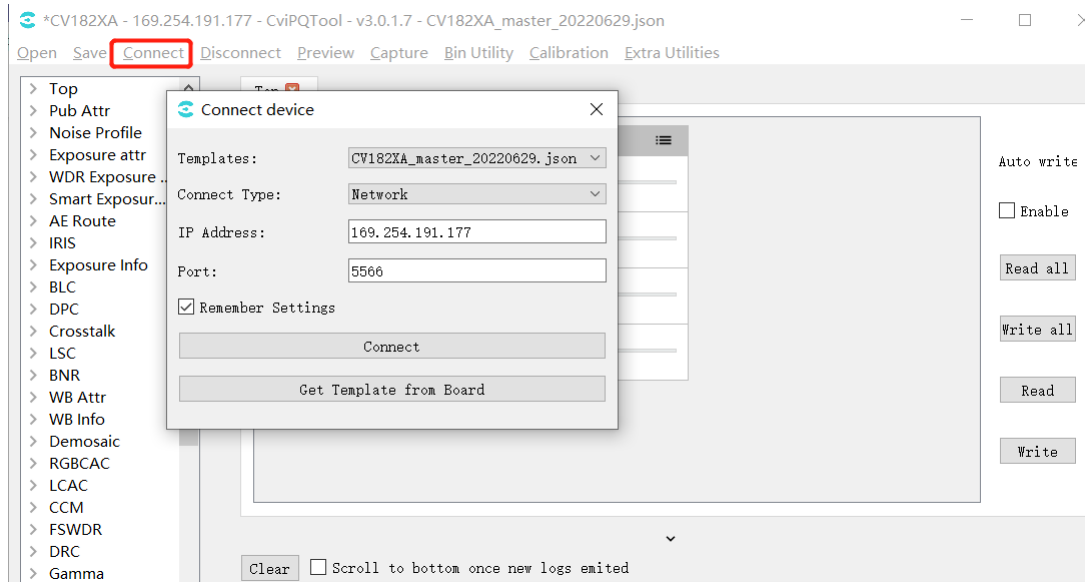


图 2.7: Connect 窗口

2.4.3.4 断开与单板的连接

在工具连接接单板的情况下，在工具栏中点击“Disconnect”按钮，即可中断 tool 与单板的连接。

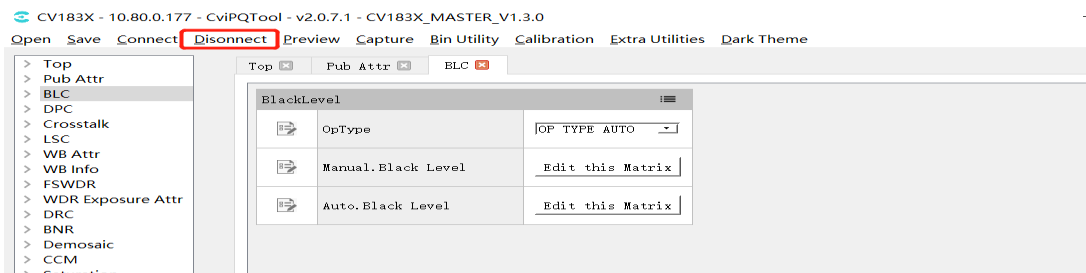


图 2.8: Disconnect 按钮

2.4.3.5 图像预览

点击工具栏“Preview”按钮，弹出 Preview 窗口，点击“Get Single Image”按钮，将抓取并显示图像。（工具与单板连接，单板连接 VLC 或显示屏正常出图）

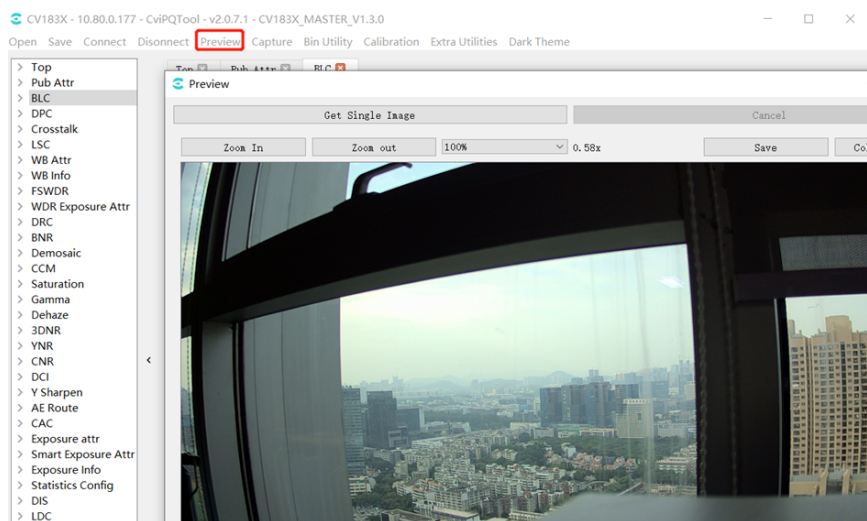


图 2.9: 图像预览窗口

2.4.3.6 抓拍工具

点击工具栏“Capture”按钮，弹出如图 Capture Tool 窗口，详细介绍见3.4.1. 抓拍工具使用说明 章节。

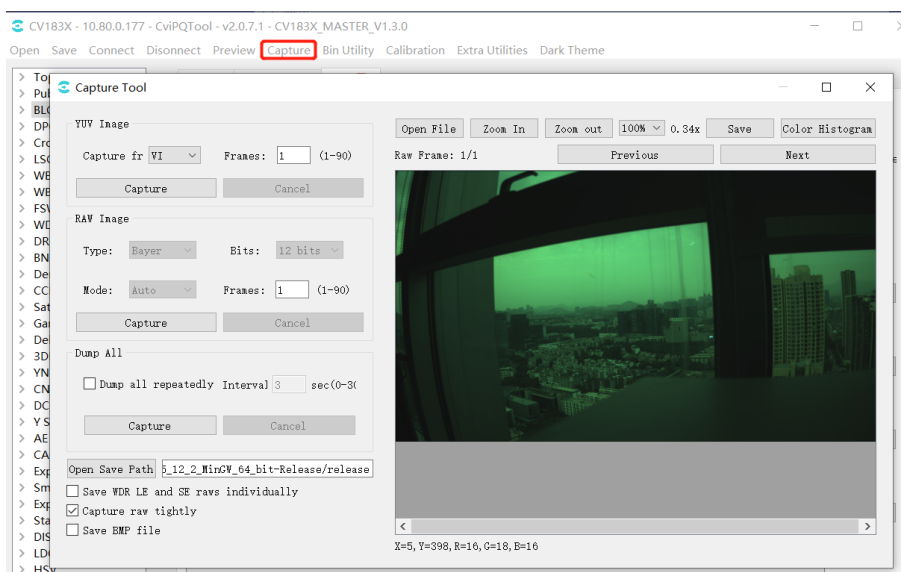


图 2.10: 抓拍工具窗口

2.4.3.7 Bin 导入导出

点击工具栏 “Bin Utility Tool” 按钮，弹出如图 Bin Utility Tool 窗口。

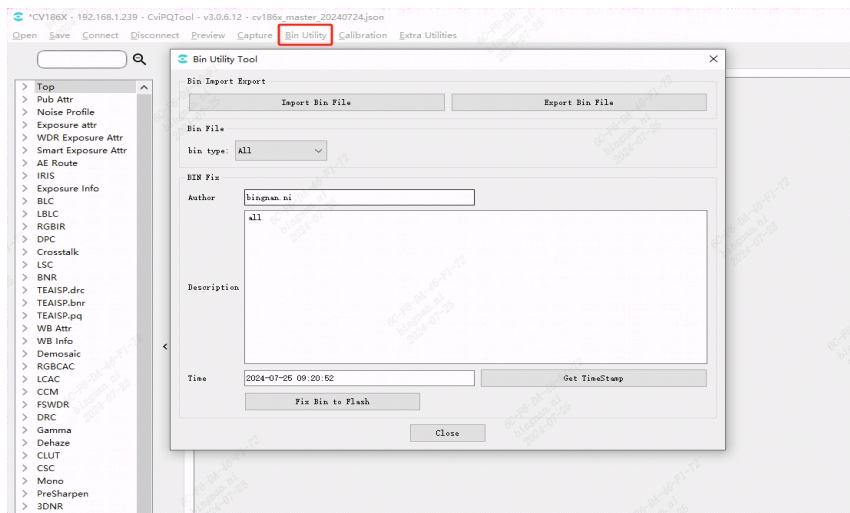


图 2.11: Bin 导入导出窗口

Export Bin File: 将板端参数导出到 pc 端保存为 bin 文件;

Import Bin File: 将 pc 端 bin 文件导入板端;

Fix Bin To Flash: 在板端生成 bin 文件;

Author,Description,Time 为文件描述信息，板端运行加载 bin 时，串口会打印。

2.4.3.8 标定工具

点击工具栏 “Calibration” 按钮，弹出如图标定工具窗口，详细介绍见3.2. ISP 标定工具 章节。

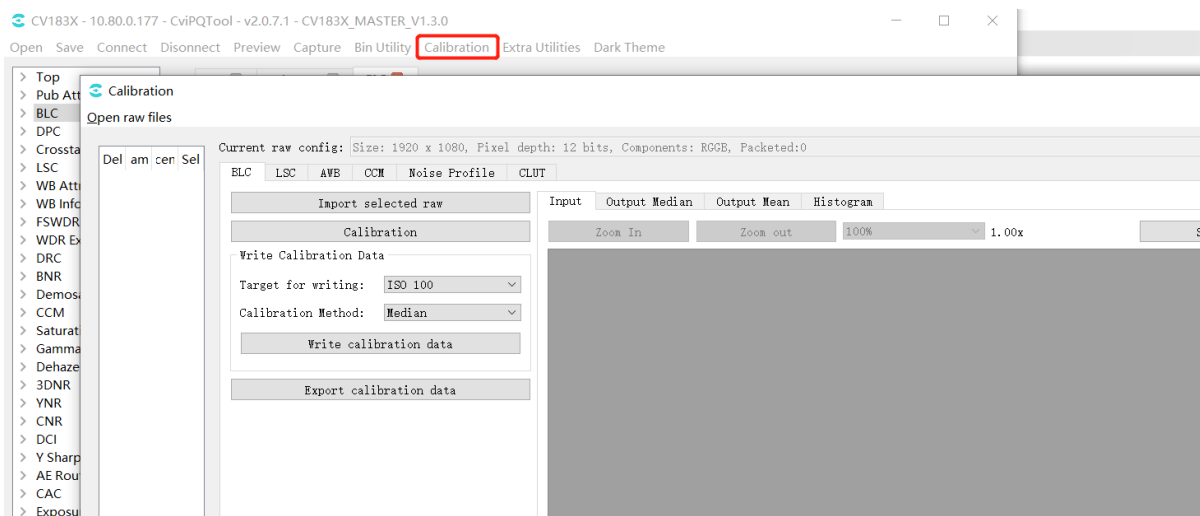


图 2.12: Calibration 工具窗口

2.4.3.9 辅助工具

点击工具栏“Extra Utilities”按钮，弹出下拉选项，可选 3A Analyser, Focus Assistant, Bracketing, Continuous Raw, DRC Tone Viewer 工具，详细介绍见3.4. 其他辅助工具使用说明 章节。

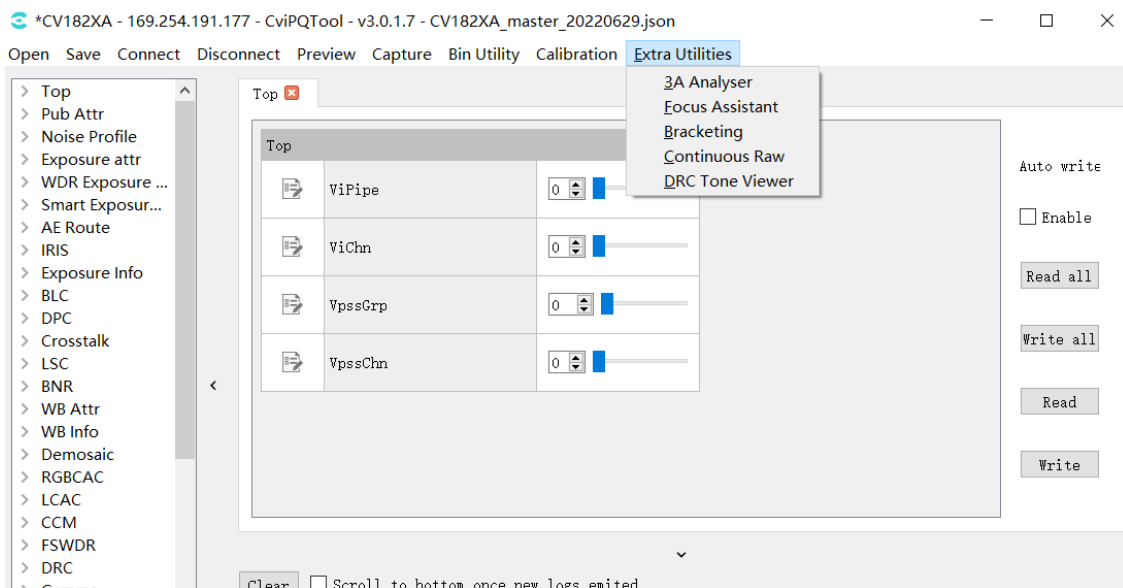
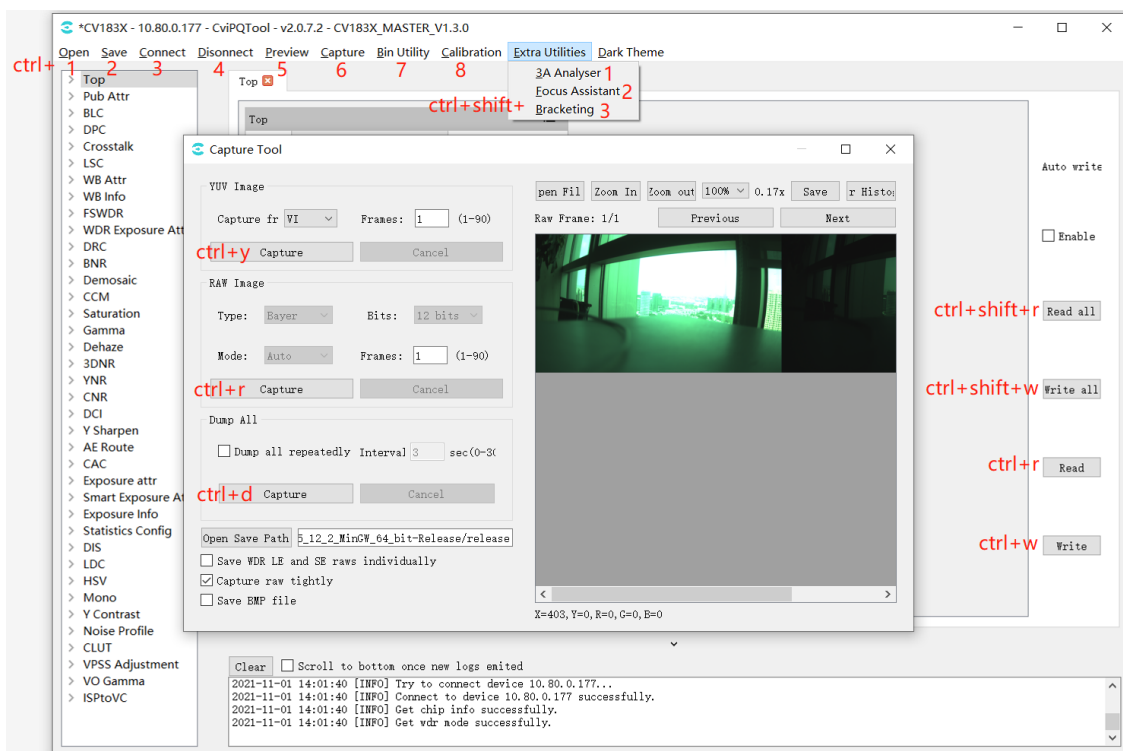


图 2.13: 辅助工具窗口

2.4.3.10 快捷键

快捷键如下图所示：



3 界面及功能说明

3.1 在线调试界面及功能说明

3.1.1 调试结构界面

打开 CviPQ Tool 工具进行联机以后，工具左侧的调试表面板显示所有模块的可调试项，结构如图。

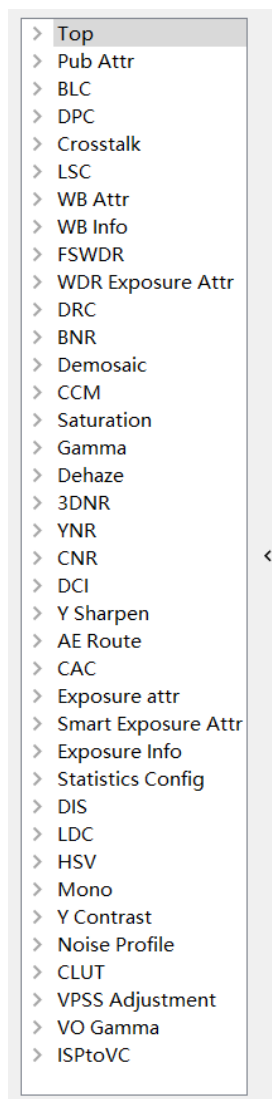


图 3.1: 工具左侧结构

点击这些模块的可调试项，在工作右侧的调试区域会显示其模块对应的调试页面，提供用户调节参数，如图 所示。

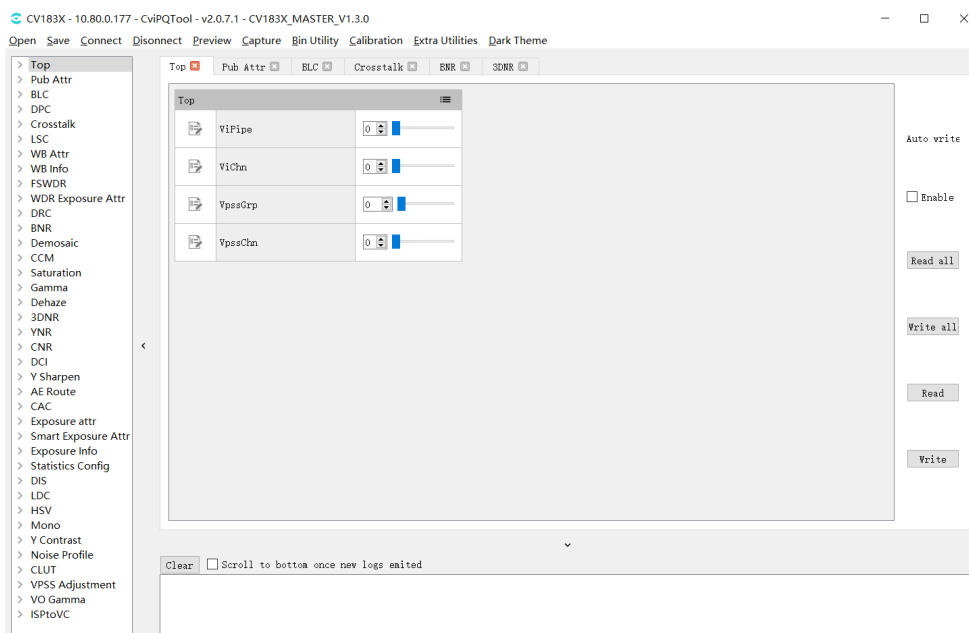


图 3.2: 调试页面

3.1.2 寄存器/算法参数调整

在工具接口左侧列表中的每项为“寄存器/算法参数”调试页。这些页对应的调试界面如图 所示，每个模块对应的调试页面中所有参数会根据不同的功能被分为若干个群组。

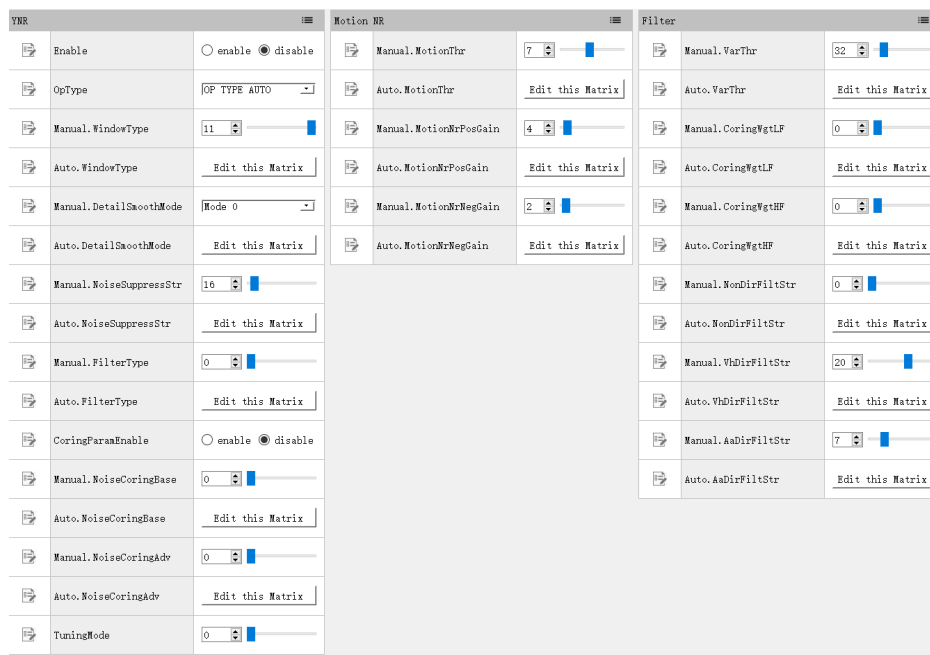


图 3.3: 寄存器/算法参数的调试页面 (以 YNR 为例)

3.1.2.1 寄存器/算法参数的查看与修改






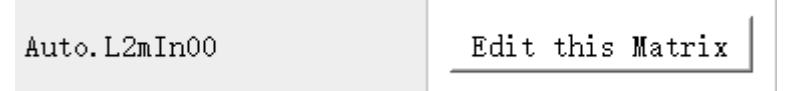
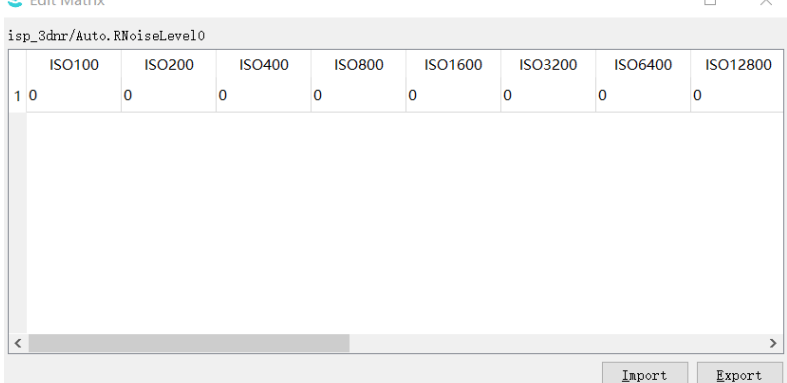
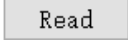
每个群组内都包含寄存器与算法参数，其中标记“”的参数是可读写参数，标记“”的参数是只读参数。参数按照其属性和取值范围不同，分为以下几类，以不同形式的控件来呈现，如表所示。

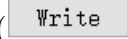
表 3.1: 寄存器/算法参数的调试项类型

类型	说明	控件操作
布尔	二选一	通过单选框设置和查看: 
数值	实数，具有一定的取值范围	使用下方控件来调整和查看值:  调整数值可以通过以下方式: 1. 直接在文本框输入数值 2. 点击文本框右侧小箭头上下微调 3. 拖动右侧滑动条
枚举	多选一	通过下拉框设置和查看值: 
矩阵	多字节序列	调试页面会显示一个编辑矩阵的按钮，如果是只读的矩阵，则会显示查看矩阵的按钮:  点击按钮会弹出一个对话框，对话框中以表格形式显示对应的矩阵值，用户可以修改表格中的值，若是只读矩阵则只能查看。 用户可以点击 Import/Export 导入导出矩阵的值，只读矩阵则只有 Export 按钮。 

3.1.2.2 板端读取数据

在工具已经连接板端的情况下，点击每个模块页面最右侧的 read 按钮 (), 工具将会读取该模块页面每项参数在当前板子上的数值，并更新到界面上显示。

3.1.2.3 将参数写入到板端

在工具已经连接板端的情况下，点击每个模块页面最右侧的 write 按钮 (), 工具将会把该模块页面上的每个可写参数写到板子端。

3.1.2.4 参数的暂存与恢复

工具对每个调试群组提供四组数据暂存空间，用户可以利用” To Set” 按钮保存参数的调整值，再利用” Load Set” 按钮进行还原和比较操作，详细操作说明如图 2-4 所示，请搭配下列 1~3 文字说明。

1. 当点此图示时，可将参数暂存与恢复的快速按钮显示出来
2. 点选” To Set” 按钮可将当前的参数暂存下来，可将不同的数值暂存在不同 Set 以便之后快速恢复。
3. 点选” Load Set” 按钮可将之前暂存的数值恢复。以图 4 为例，使用者已暂存” Set1” 和” Set2” 两组数值，就可用” Load Set1” 或” Load Set2” 按钮快速恢复和比较。

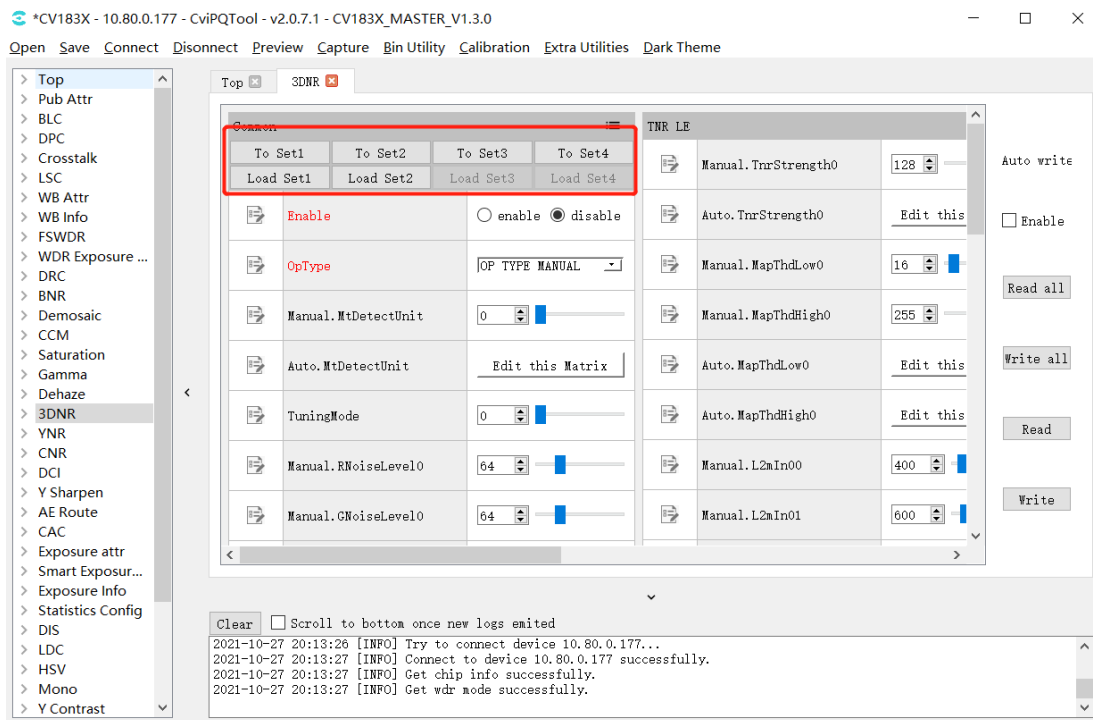


图 3.4: 参数暂存操作接口 (以 3DNR 为例)

3.1.3 调试数据的批量读写与自动写入

当打开调试工具进入主接口时，可在最右侧看到数据的读写控制接口，如图 3.5 所示。

Auto write

☐ Enable

Read all

Write all

Read

Write

图 3.5: 数据读写控制接口

读写控制台的操作需要先将工具连接单板，具体的功能有：

- Read All 按钮：点击此按钮，工具会读取所有模块页面的参数项在板子端的当前数值。建议打开工具进行调试工作前或者导入 BIN 档后，执行一次全部读取的操作。
- Write All 按钮：点击此按钮时，工具会将所有模块页面参数项的值写入到板子端。
- Auto Write 自动写入开关：当勾选此项时，每次调整一个可写的参数项，工具都会将参数的新值写入到板子端。建议调试时打开此项，保证调整值能立刻生效。
- Read 按钮：点击此按钮时，工具会将当前打开的模块页面所有参数项在已连接板端的当前值读回到接口上。
- Write 按钮：点击此按钮时，工具会将当前打开的模块页面所有参数项的值写入到板端。

3.1.4 曲线可视化调试

3.1.4.1 通用曲线功能说明

工具中通用曲线如图 3.6 所示，如在 3DNR 调试页面中，点击 Edit this Curve 按钮可进入曲线窗口。

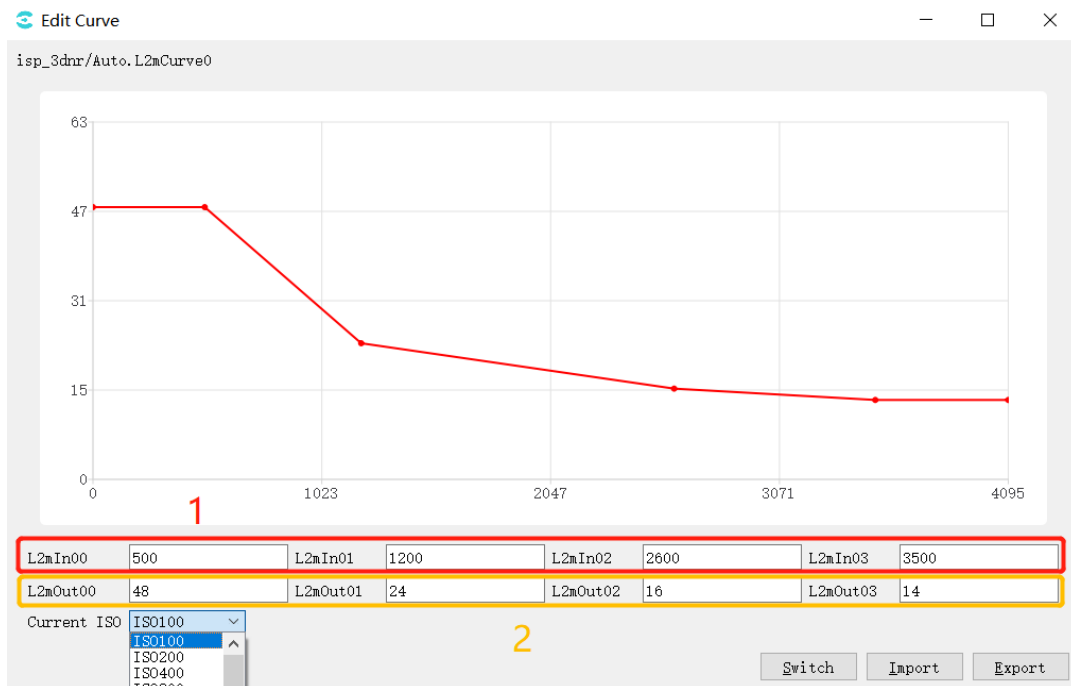


图 3.6: 3DNR 页面曲线操作示例图

图中 1 为曲线横坐标，2 为曲线纵坐标，对应曲线中 4 个点，修改它们的值，曲线上的点随之改变；也可通过鼠标拖动曲线上的点，对应横、纵坐标值随着改变。

Current ISO 下拉选项，可选择不同 ISO 值，查看其对应曲线。

Switch 模式切换按钮，在曲线模式下点击，进入如下表格模式，表格前 4 行为横坐标值，后 4 行为纵坐标值，表格可编辑。

	ISO100	ISO200	ISO400	ISO800	ISO1600	ISO3200	ISO6400	ISO12800	ISO25600	ISO51200	ISO102400	ISO204800	ISO409600	ISO819200	ISO1638400	ISO3276800
1 1111	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500	500
2 1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200
3 2600	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600	2600
4 3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500
5 63	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48
6 24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24
7 16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
8 14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14

Import，从文件中导入参数。

Export，将参数值导出到文件。

3.1.4.2 Gamma 可视化调试

在调试工具左侧清单点击 gamma 页面，进入 Gamma 可视化调试页面，如图 所示。

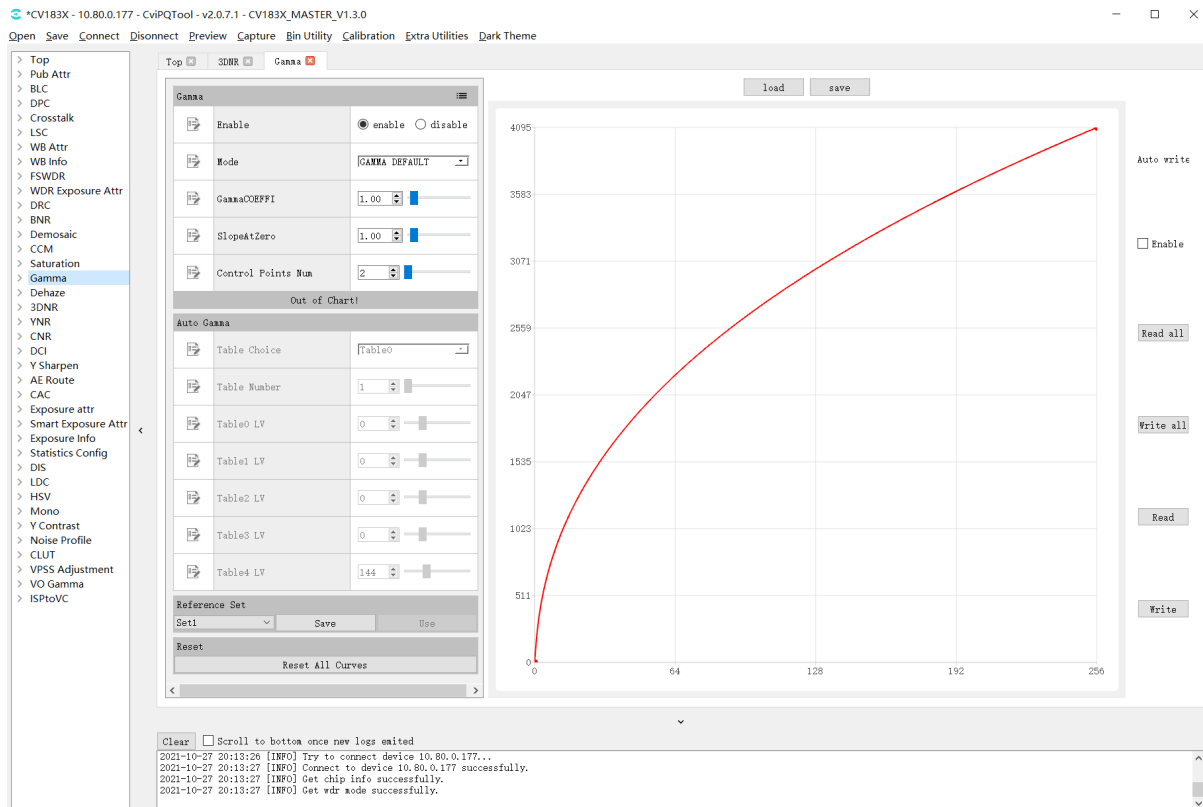


图 3.7: Gamma 调试页面

以下是具体的参数项说明和用法：

- Gamma
 - Enable 参数项，设置这项为 enable 状态使能 Gamma。
 - Mode 参数项，代表曲线调整模式，当选择 GAMMA_DEFAULT/GAMMA_sRGB，曲线会重置为默认的曲线并且用户不可手动调整曲线；当选择 GAMMA_USER_DEFINE/GAMMA_AUTO 时曲线会被重置为起点为 (0,0) 终点为 X/Y 轴最大值的直线，用户可以手动调整曲线走向。
 - GammaCOEFFFI 参数项，通过在文本框输入数值或者拖动滑动条，可以将当前曲线恢复为标准的 Gamma 曲线（系数范围为 0.01 到 20.0）。
 - SlopeAtZero 参数项，通过在文本框输入数值或者拖动滑动条，可以调节 Gamma 曲线在零点的斜率值（斜率范围为 0.01 到 20.0）。
 - Control Points Num 控制点，当选择 GAMMA_USER_DEFINE/GAMMA_AUTO 时，通过调整此项参数，用户可以设置能操作的控制点数量。
- Auto Gamma（gamma mode 设置为 GAMMA_AUTO 有效）
 - Table Choice，根据 light value 选择 gamma table 0~5，目前使用前 4 组；
 - Table Number，当前使用的 table 数

- Table0~4 LV, 对应 5 个 light value 阈值 (LV:-5~15)

- Reference Set

曲线数据暂存区，工具提供 set1 到 set3 共三组暂存空间，当用户点击 save 按钮将当前曲线的数据保存到选定的 set，当用户点击 use 按钮可将已暂存的数据恢复到曲线接口上。默认未进行暂存操作时，use 按钮为灰色不可点击。

- Reset

Reset 为重置按钮，点击后会重置曲线为起点为 (0,0) 终点为 X,Y 为最大值的直线。

- Load&Sav

- 点击“Load”按钮，可以从文件中读取曲线数据，写回到工具数据缓存区，同时刷新显示曲线。
- 点击“Save”按钮，可以将当前曲线数据保存到本地的文本文件。工具支持保存两种格式的数据文件：Cvitek Json file、Other text file

- 点击页面读写控制接口的 Read 按钮，工具会从板端读取曲线数据，并刷新曲线的显示。
- 点击 Write 按钮，工具会将当前曲线的数据写入到板端。
- 如果已勾选 Auto Write，当调整曲线时，曲线的数据会自动写到板端生效。

3.1.4.3 VO Gamma

在调试工具左侧清单点击 VO Gamma，进入 VO Gamma 可视化调试页面，如图 所示。

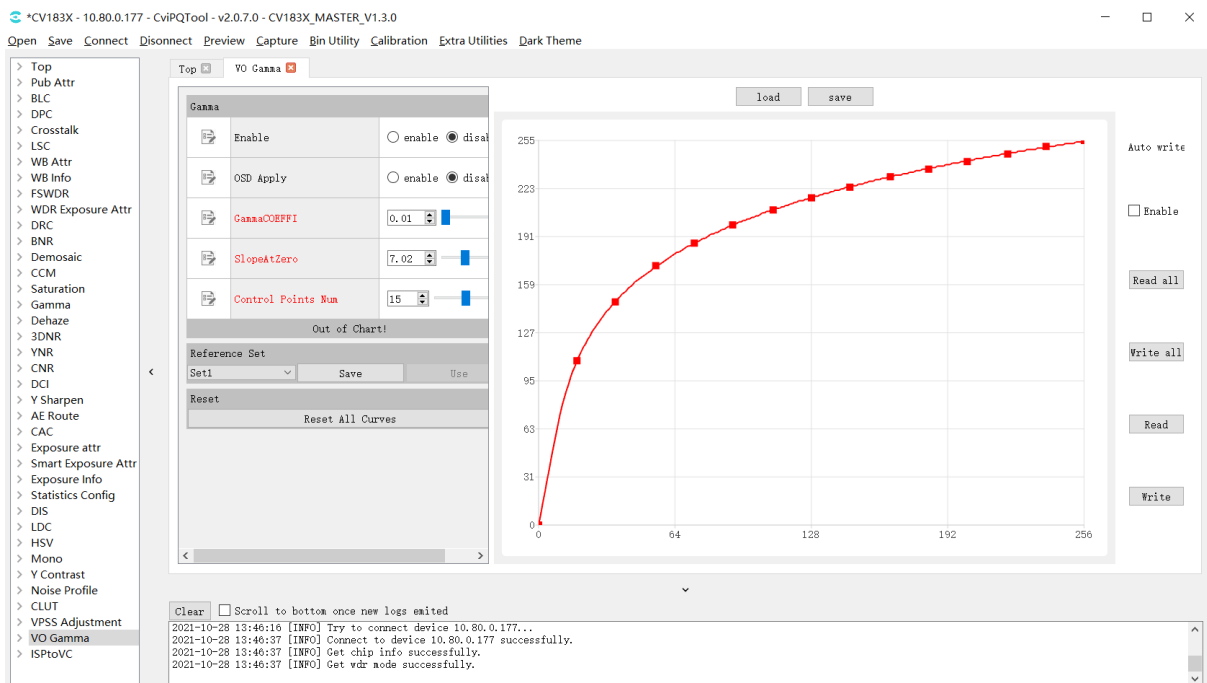


图 3.8: VO Gamma 调试页面

参数项说明：

- Gamma

- Enable 参数项, 设置这项为 enable 状态使能 VO Gamma。
- OSD Apply: 显示水印开关。
- GammaCOEFFFI 参数项, 通过在文本框输入数值或者拖动滑动条, 可以将当前曲线恢复为标准的 Gamma 曲线 (系数范围为 0.01 到 20.0)。
- SlopeAtZero 参数项, 通过在文本框输入数值或者拖动滑动条, 可以调节 Gamma 曲线在零点的斜率值 (斜率范围为 0.01 到 20.0)。
- Control Points Num 控制点, 当选择 GAMMA_USER_DEFINE/GAMMA_AUTO 时, 通过调整此项参数, 用户可以设置能操作的控制点数量。

- Reference Set

曲线数据暂存区, 工具提供 set1 到 set3 共三组暂存空间, 当用户点击 save 按钮将当前曲线的数据保存到选定的 set, 当用户点击 use 按钮可将已暂存的数据恢复到曲线接口上。默认未进行暂存操作时, use 按钮为灰色不可点击。

- Reset

Reset 为重置按钮, 点击后会重置曲线为起点为 (0,0) 终点为 X,Y 为最大值的直线。

- Load&Sav

- 点击 “Load” 按钮, 可以从文件中读取曲线数据, 写回到工具数据缓存区, 同时刷新显示曲线。
- 点击 “Save” 按钮, 可以将当前曲线数据保存到本地的文本文件。工具支持保存两种格式的数据文件: Cvitek Json file、Other text file

- 点击页面读写控制接口的 Read 按钮, 工具会从板端读取曲线数据, 并刷新曲线的显示。

- 点击 Write 按钮, 工具会将当前曲线的数据写入到板端。

- 如果已勾选 Auto Write, 当调整曲线时, 曲线的数据会自动写到板端生效。

用法:

1. tool 与板端连接正常, 板端连接显示屏正常输出影像;
2. 使用 “GammaCOEFFFI”, “SlopeAtZero”, “Control Points Num” 功能调整出需要的曲线;
3. “Enable” 选择 enable, 点击 write;
4. 观察显示屏, 影像亮度会有变化。

3.1.5 曝光数据状态和参数设置

3.1.5.1 曝光数据状态

Exposure Info 页面可显示当前的曝光数据状态, 包含曝光时间、增益、ISO... 等数据, 如图 所示。

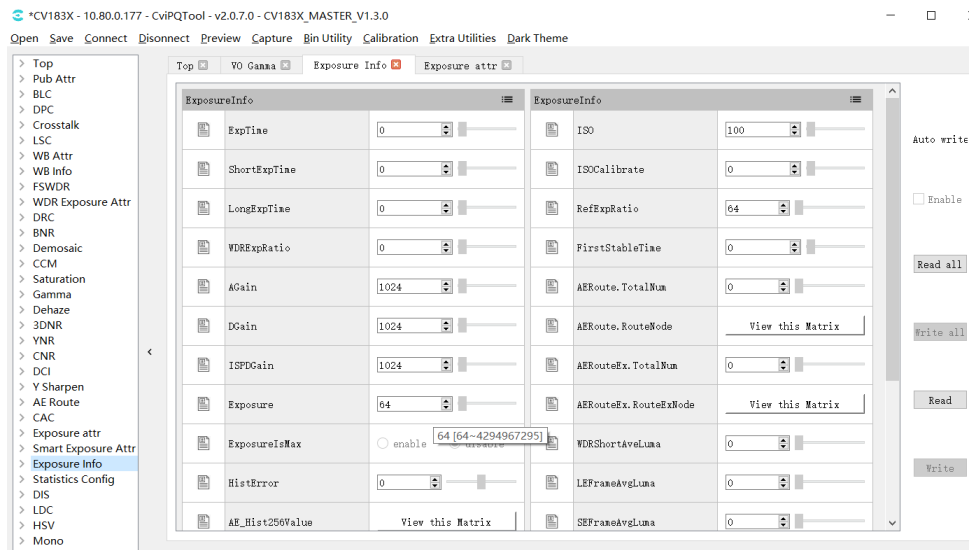


图 3.9: 曝光数据状态页面

3.1.5.2 曝光参数设置

Exposure Attr 页面可设置曝光相关参数，包含曝光时间、增益、ISO…等参数，如图 所示。

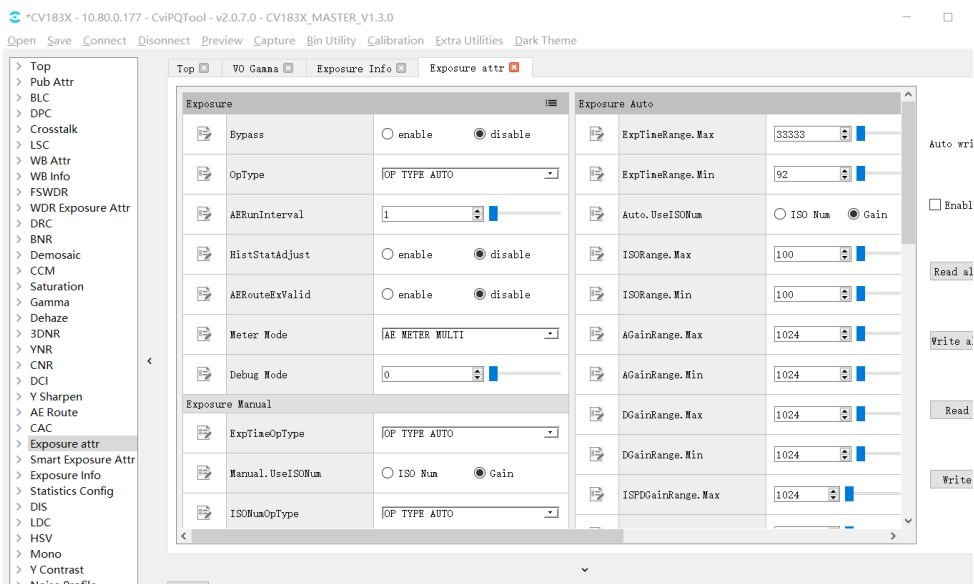


图 3.10: 曝光参数设置页面

3.1.6 白平衡数据状态和参数设置

3.1.6.1 白平衡数据状态

WB Info 页面可显示当前的白平衡数据状态，包含增益、色温…等数据，如图 所示。

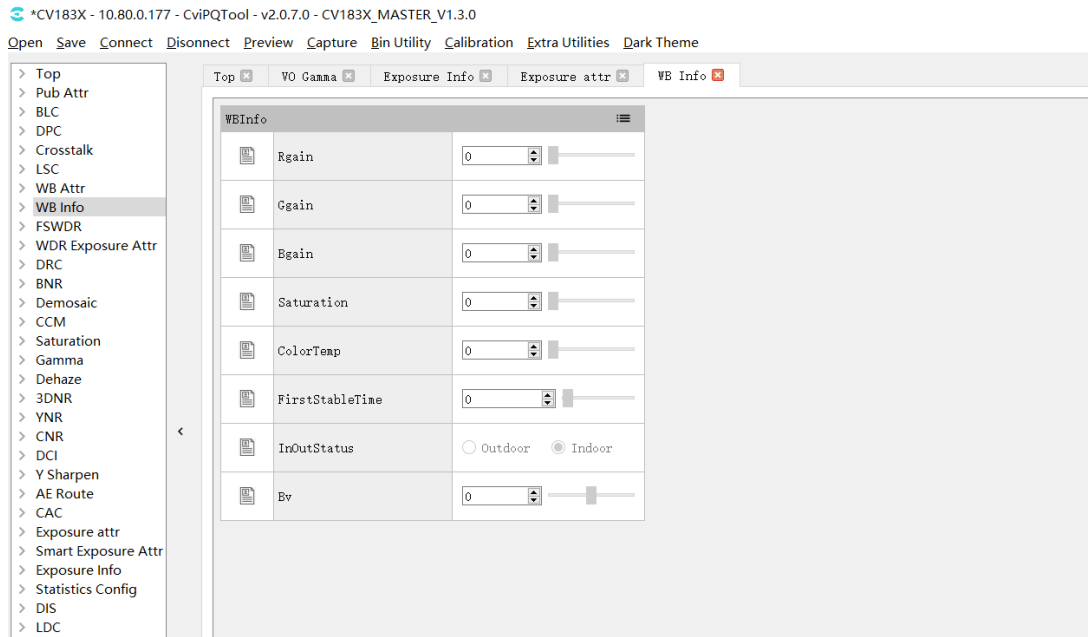


图 3.11: 白平衡数据状态页面

3.1.6.2 白平衡参数设置

WB Attr 页面可设置白平衡相关参数，包含增益、色温…等参数，如图 所示。

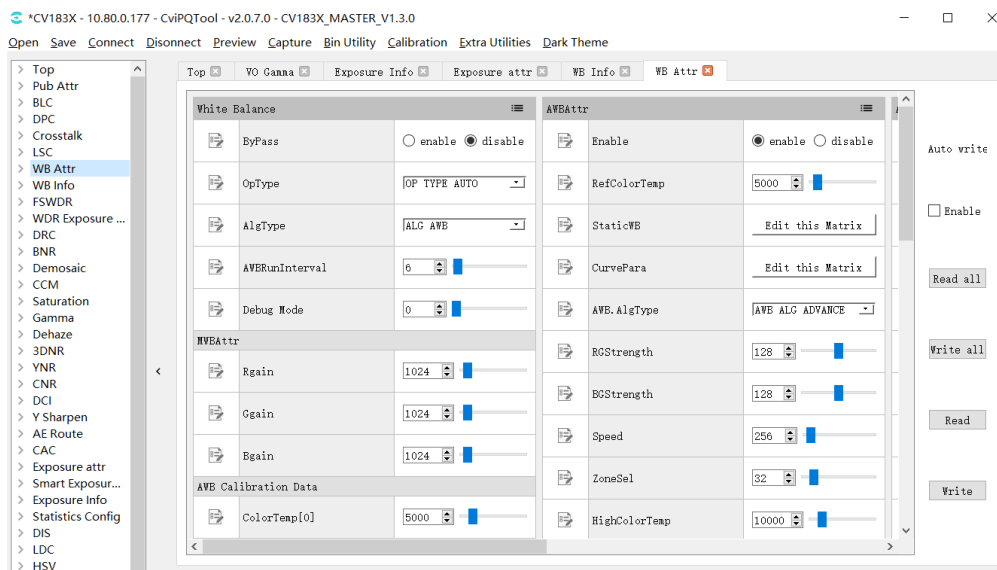


图 3.12: 白平衡参数设置页面

3.1.7 WDR 曝光参数设置和 Local-tone 开关

详细 WDR 和 DRC 参数说明请参考”图像调优指南” WDR 章节和 DRC 章节。

3.1.7.1 WDR 曝光参数设置

WDR Exposusre Attr 页面可设置宽动态曝光相关参数，包含曝光比…等参数，如图 所示。

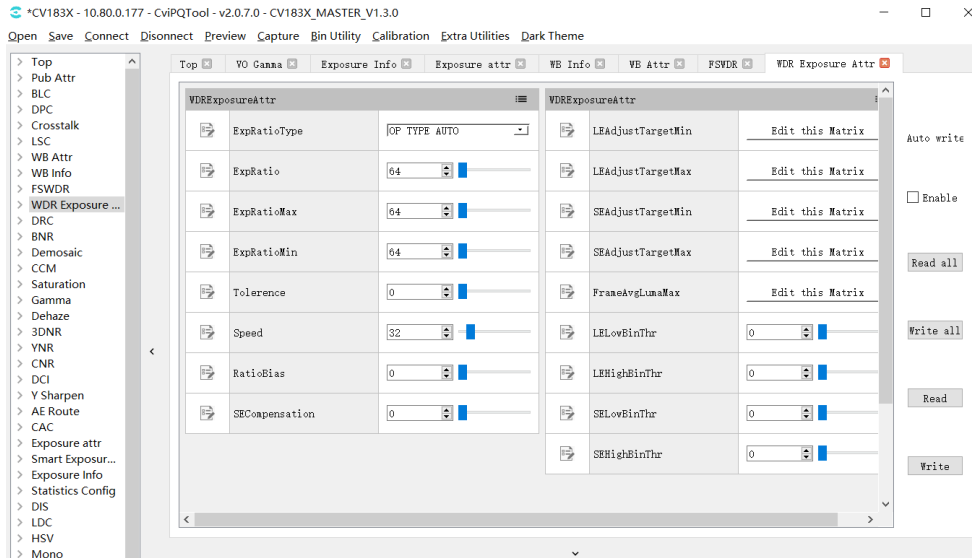


图 3.13: WDR 曝光参数设置页面

3.1.7.2 Local-tone 开关

在 DRC 页面可开启或关闭 Local-tone，当 [LocalToneEn] 参数设置 enable 时表示 Local-tone 开启，当 [LocalToneEn] 参数为设置 disable 时表示 Local-tone 关闭，如图 所示。

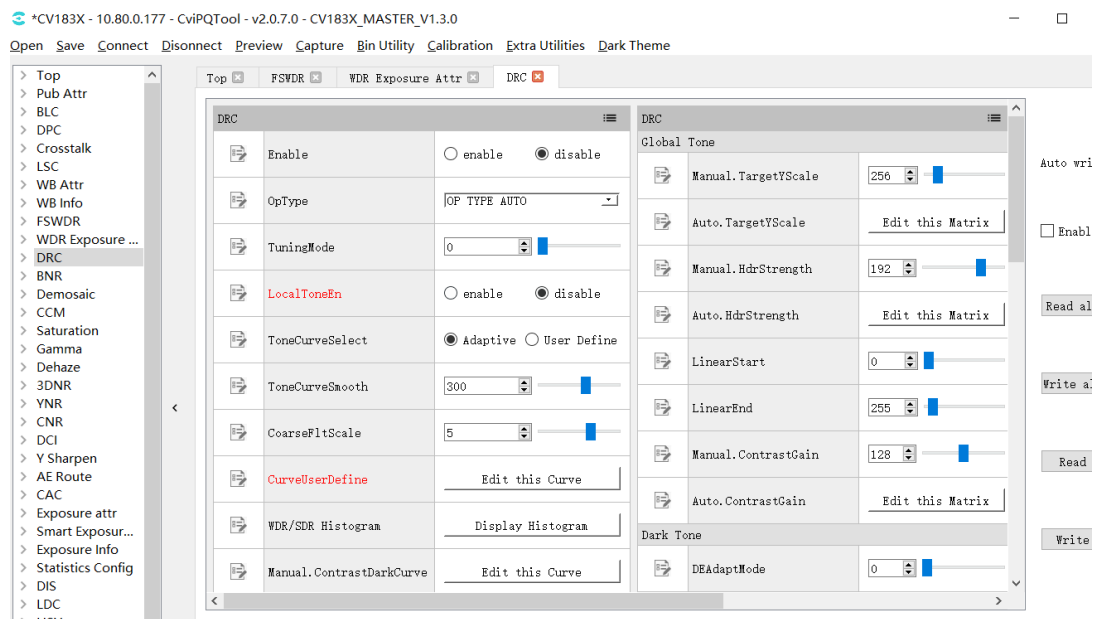


图 3.14: DRC Local-tone 开关

3.1.8 3DNR 和 YNR 降噪调试

降噪相关调试参数位于 3DNR 和 YNR 页面，详细降噪参数说明请参考”图像调优指南” 3DNR 章节和 YNR 章节。

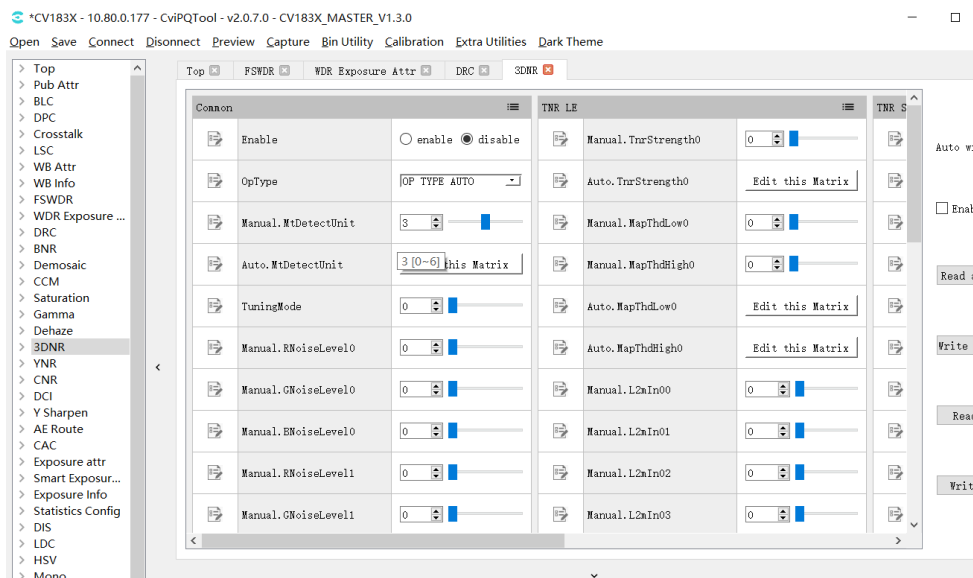


图 3.15: 3DNR 页面

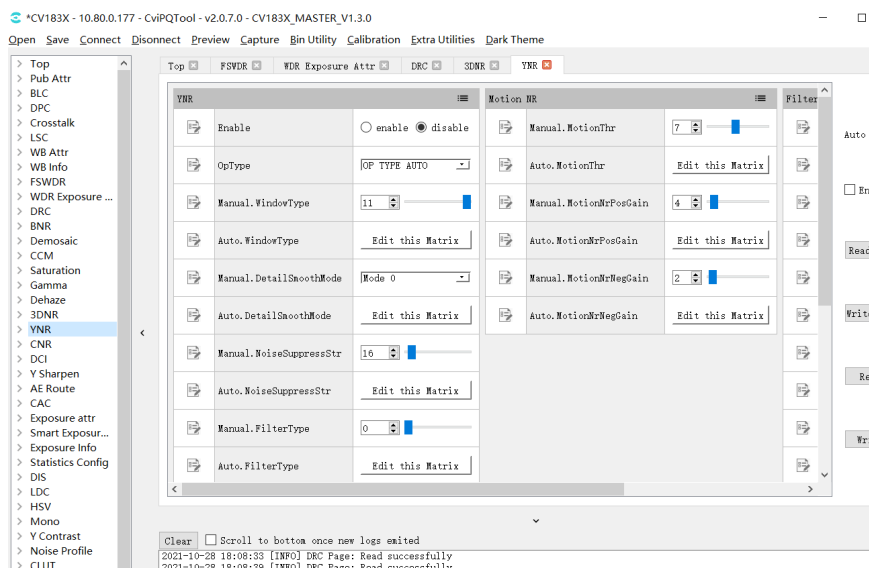


图 3.16: YNR 页面

3.1.9 Y Sharpen 锐化和边缘增强调试

Y Sharpen 页面包含锐化和边缘增强相关参数，详细锐化和边缘增强参数说明请参考“图像调优指南”Sharpen 章节。

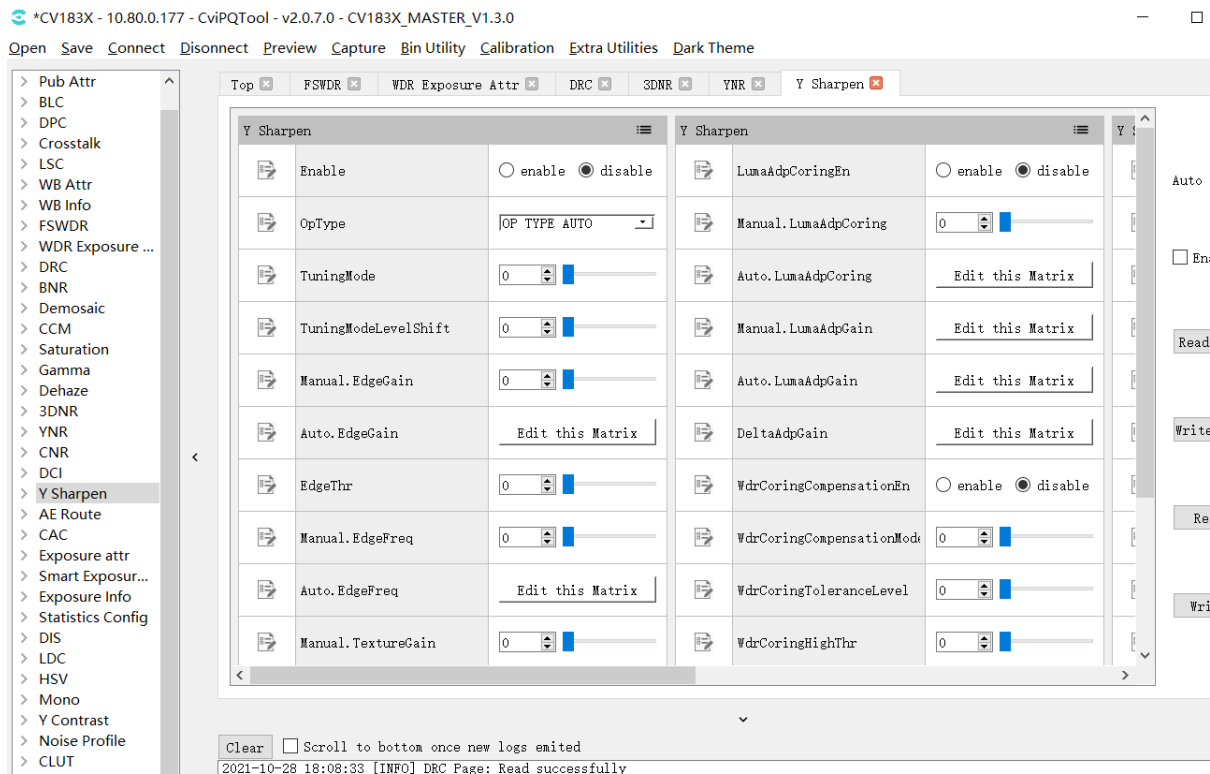


图 3.17: Y Sharpen 页面

3.1.10 Top 页面

Top 页面包含 ViPipe 和 ViChn 参数，如果版端有两个 sensor，可用 Vipipe 切换不同 Sensor 的调试参数，另外可用 ViChn 切换屏幕上要显示那个 sensor 的影像。

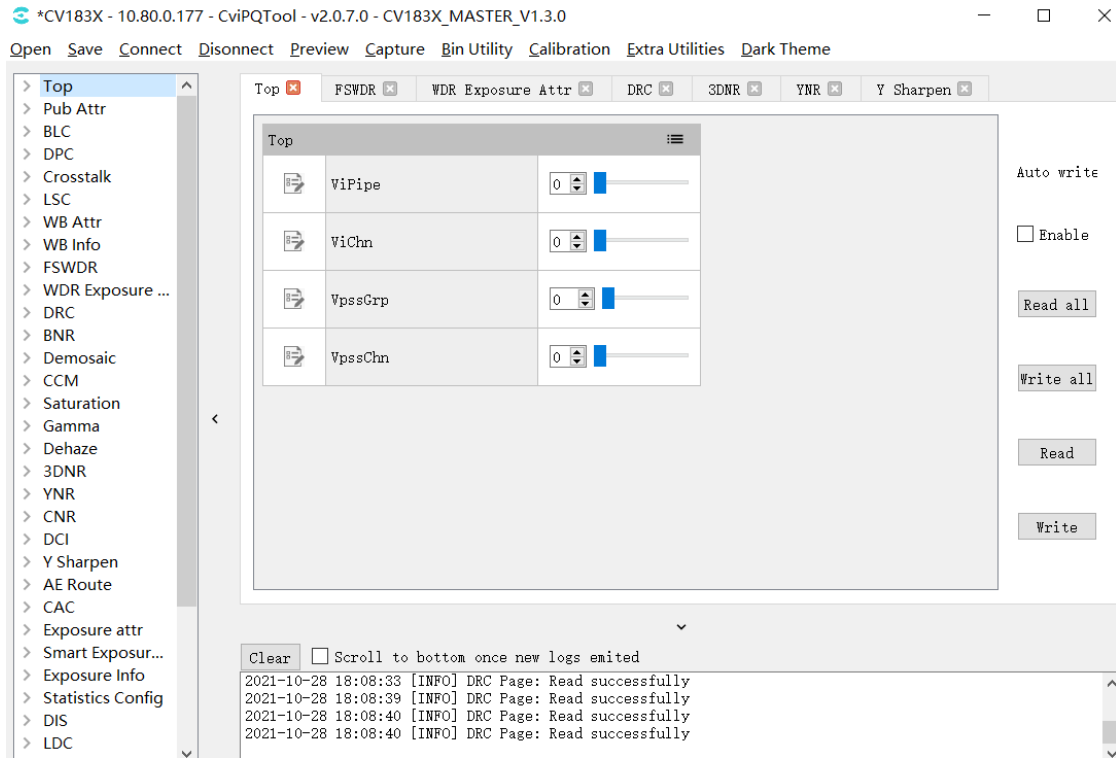


图 3.18: Top 页面

3.1.11 VPSS Adjustment 页面

VPSS Adustment 页面用于调试 vpss 相关参数，调试过程有以下几点需要注意：

1. 使用 pqtool 调试过程中 scene id 和 vpss grp id 是一一对应的，即 scene0 的参数会写到 grp 0；
2. App 实际使用时需要在每次 load pq bin 后主动调用 CVI_VPSS_SetGrpParamfromBin 设定 scene 与 vpss grp id 的绑定关系，设定的同时 pq bin 中的 vpss 参数才会生效，否则使用 pqtool 调试好的 vpss 参数在 load pq bin 后是不会生效的。

注：这样设计的原因是 pq 参数调试过程中无法预测 app 实际会使用哪个 vpss grp id，所以调试时是一一对应调试，app 使用调试生成的 pq bin 时需要根据实际场景指定映射关系。

3.1.12 LDC 页面

LDC 页面用于调试 VI 和 VPSS LDC module 相关参数，在使用过程有以下几点需要注意：

1. 在 Read all 和 Write all 时，不包含对 LDC 页面的读写操作。故而对 LDC 页面进行相关读写设置时，需要进入此页面进行单独操作（点击 Read 或 Write 按钮）方可有效。
2. 目前 LDC 运算操作是在板端进行且其计算量大，耗时较长，完成一次计算操作需要约 2 分钟，PQtool 在此间需要等待，之后才能对 PQtool 进行下次的有效操作。

3.2 ISP 标定工具

请参考“图像调优指南”完成 ISP 标定。

3.3 高级功能

3.3.1 参数的导出导入

工具的参数可以支持在 PC 端以配置文件的方式导入导出，也支持工具将参数固化到板端。

- 在 PC 端导入导出参数到配置文件的操作，请参考[2.4.3.1](#) 和[2.4.3.2](#) 打开调试数据文件章节。
- 将参数固化到板端，或者从板端将参数导出并备份到 PC 本地文件，请使用 BIN utility 工具进行操作。

3.3.2 通讯日志

与单板连接后，所有与板端的数据交互都会记载日志并显示在通讯日志窗口，如图 所示。与板端进行数据交互时，记载的日志包含以下信息：

- 通讯时间；
- 通讯方式与参数；
- 通讯内容（如当前读取或写入的调试项）；
- 如果通讯出错，则显示错误信息。

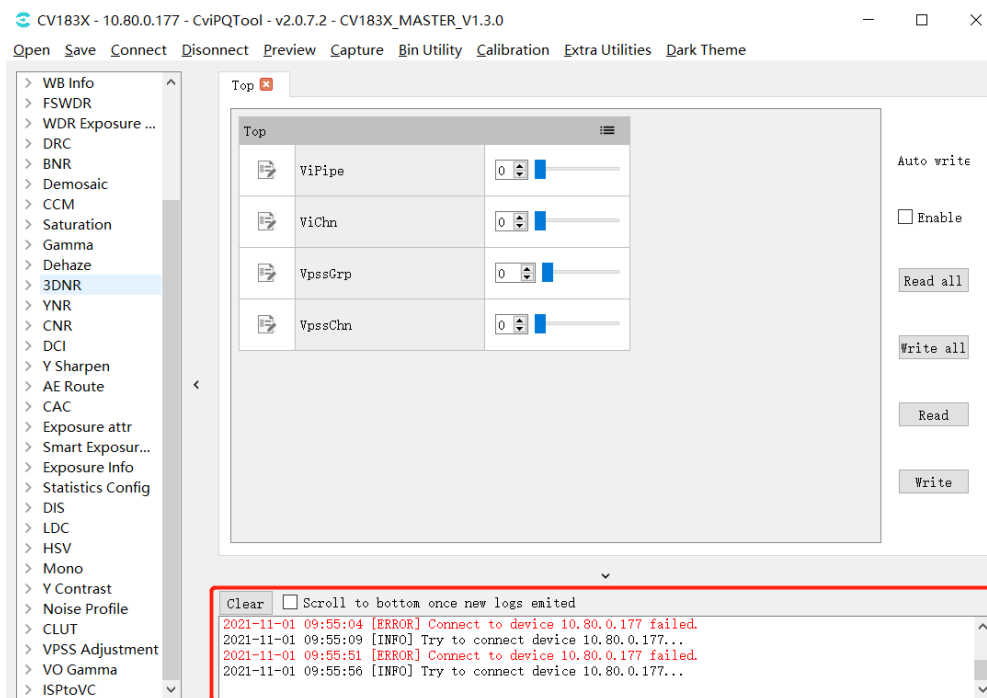


图 3.19: 通讯日志窗口

3.4 其他辅助工具使用说明

3.4.1 抓拍工具使用说明

可以抓取板端的图像数据（YUV 和 Raw）并保存为本地文件。

3.4.1.1 工具界面

在调试工具菜单栏点击 Capture 按钮即可打开抓拍工具窗口，如图 所示。

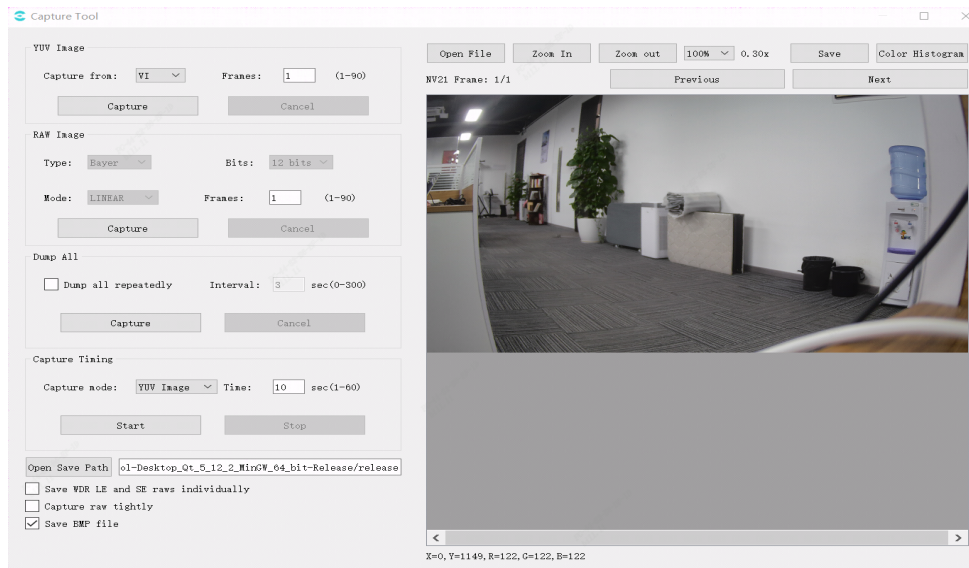


图 3.20: 抓拍工具窗口

Save WDR LE and SE raws individually: 勾选时, LE raw,SE raw 分开存放; 去勾选时, 两张 raw 合并存放。

Capture raw tightly: 抓取 raw 的方式不同, 勾选时抓取速度快点, 去勾选时速度慢些。

Save BMP file: 勾选将保存一张 bmp 图像, 默认保存 jpg 和 bmp 格式图片。

image 窗口下方, 显示对点的 x,y 坐标及 r,g,b 值, 会随着鼠标移动改变。

Dump all repeatedly: 勾选将按 Interval 设置时间 (s) 重复 dump all。

3.4.1.2 抓取 YUV 图像数据

抓取 YUV 数据的步骤:

1. 在“YUV Image”分组框内选择要抓取 YUV 数据的来源位置, 即在 Capture from 下拉菜单选择 VI/VPSS;
2. 在 Frames 文本输入框输入要抓取的帧数;
3. 点击 Capture 按钮, 若抓取成功会自动将 YUV 数据储存在“Open Save Path”所显示的路径位置, 且 YUV 数据会显示在右侧图像显示区域, 可通过点击 Previous/Next 按钮逐帧查看;

——结束

3.4.1.3 抓取 Raw 图像数据

抓取 Raw 数据的步骤：

1. 在“Raw Image”分组框内选择要抓取 Raw 数据的比特位数，即在 Bits 下拉菜单选择 8bits/10bits/12bits/16bits；
2. 在 Mode 下拉菜单选择当前模式为 Linear、WDR 或 Auto；
3. 在 Frames 文本输入框输入要抓取的帧数；
4. 点击 Capture 按钮，若抓取成功会自动将 Raw 和 Raw Info 数据储存在“Open Save Path”所显示的路径位置，且 Raw 数据会显示在右侧图像显示区域，可通过点击 Previous/Next 按钮逐帧查看；

——结束

3.4.1.4 Dump All

Dump All 可抓取 YUV、RAW 图像及日志信息，抓取文件如下：

```
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_.jpg
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_.json
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_.raw
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_.txt
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_.yuv
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_-ae.bin
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_-aelog.txt
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_-awb.wbin
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_-awblog.txt
```

抓取步骤：

1. 在 YUV Image 中设置好 yuv 抓取信息；
2. 在 RAW Image 中设置好 raw 抓取信息；
3. 点击 Dump All 中 Capture 按钮，tool 将先抓取并保存 raw 图像及文件，然后抓取并保存 yuv 图像及文件，右侧图像显示区将显示 raw 图像，可通过点击 Previous/Next 按钮逐帧查看。

3.4.1.5 Caputure Timing

定时抓取 YUV、RAW、Dump All 三类图片，抓取步骤如下：

1. “在 Capture mode”的下拉菜单选中要抓取的图片类型，默认为抓取 YUV 图片。
2. 在“Time”中输入要定时抓取的时间，默认为 10 秒。
3. 点击“Start”按钮，即可启动定时，待达到定时时间时，就会从板端获取对应相关数据。
4. 若在定时抓取的中间，想取消此操作，点击“Stop”按钮即可。

3.4.1.6 YUV 图像和 Raw 图像文件名格式说明

YUV 图像文件名格式范例如下：

```
1920X1080_RGGB_Linear_20211101105807_00_-color=3_-bits=12_-frame=1_-hdr=0_ISO=110_.yuv
```

-1920X1080 表示宽度为 1920 高度为 1080

-color=3 表示 BayerID=3 (0: BGGR, 1: GBRG, 2: GRBG, 3: RGGB)

-bits=12 表示影像有效位数是 12-bit

-frame=1 表示此档案包含 1 张 YUV 图像

-20211101105807 表示存盘的时间，格式为” yyyymmddhhmmss”

RAW 图像文件名格式范例如下：

```
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_.raw
```

-3840X1080 表示宽度为 3840 高度为 1080

-RGGB 表示 BayerID

-WDR 表示为宽动态图像

-color=3 表示 BayerID=3 (0: BGGR, 1: GBRG, 2: GRBG, 3: RGGB)

-bits=12 表示影像有效位数是 12 bits

-frame=1 表示此档案包含 1 张 RAW 图像

-hdr=1 表示为宽动态影像

-20211101111620 表示存盘的时间，格式为” yyyymmddhhmmss”

这里的 Raw 经过解压裁剪：size = width * height * 2 * frame；

3.4.1.7 Raw Info 文件格式和内容说明

RAW Info 文件名格式范例如下：

```
3840X1080_RGGB_WDR_20211101111620_00_-color=3_-bits=12_-frame=1_-hdr=1_ISO=100_.txt
```

RAW Info 的文件名格式和 RAW 图像的文件格式完全相同，只差在扩展名为 TXT 文本文件。

RAW Info 档案内容范例如下：

```
ISO = 100
Color Temp. = 4482
ISP DGain = 1024
Long Exposure = 12630
Short Exposure = 3157
Exposure Ratio = 4
reg_wbg_rgain = 1834
reg_wbg_bgain = 2166
reg_ccm_00 = 1736
reg_ccm_01 = -589
reg_ccm_02 = -123
reg_ccm_10 = -649
reg_ccm_11 = 2016
reg_ccm_12 = -343
reg_ccm_20 = -32
reg_ccm_21 = -742
reg_ccm_22 = 1798
reg_blc_offset_r = 241
reg_blc_offset_gr = 241
reg_blc_offset_gb = 241
reg_blc_offset_b = 241
reg_blc_gain_r = 1089
reg_blc_gain_gr = 1089
reg_blc_gain_gb = 1089
reg_blc_gain_b = 1089
```

3.4.2 视频播放工具使用说明 (VLC)

3.4.2.1 使用 VLC 播放板端视频流步骤

步骤一： VLC 播放工具下载，可到此地址下载免安装 Windows 版本的 VLC 播放软件。

其他平台的 VLC 版本请到 VLC 官网自行下载。(<https://get.videolan.org/vlc/3.0.10/win32/vlc-3.0.10-win32.zip>)

步骤二： 打开工具, 对 VLC 做参数设定, 保存后重启 VLC。

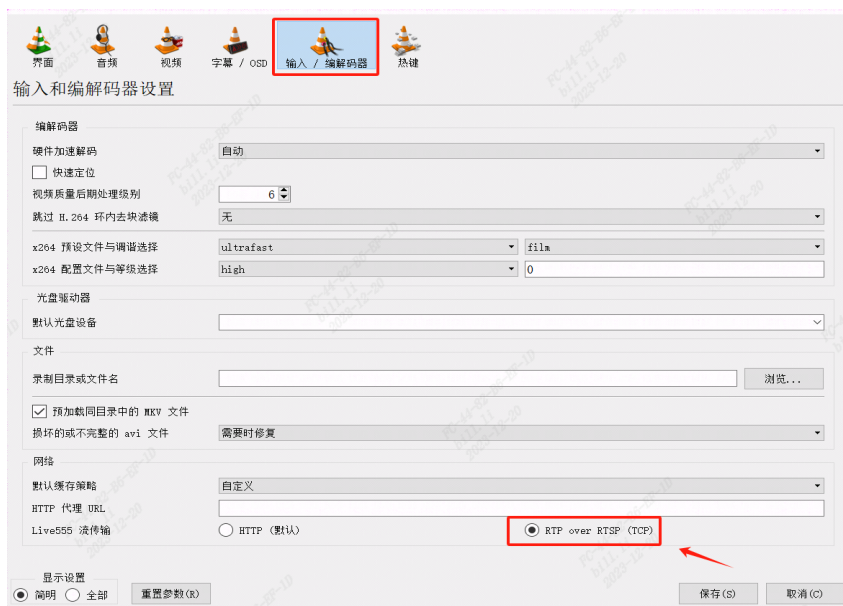


图 3.21: VLC 设置界面

步骤三： 使用网线连接开发板与电脑端，详情请参考“EVB 使用手册”。

步骤四：在板端通过串口进入文件系统，启动 CviISPTool 程序

开发板连接的电脑需要设置到相同 IP 网段，并保证以相互 ping 通后，便可在与开发板连接的电脑上启动 VLC 进行播放。如下是播放设置部分参数，设置完后点击播放按钮便可以看到 Sensor 画面。

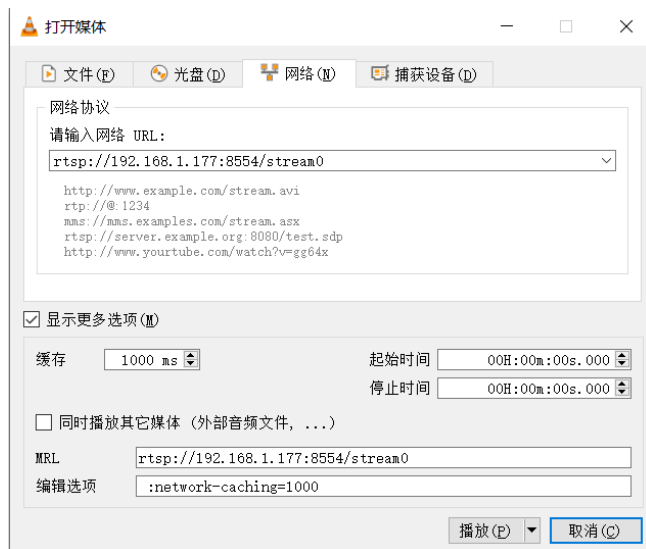


图 3.22: VLC 播放参数设置

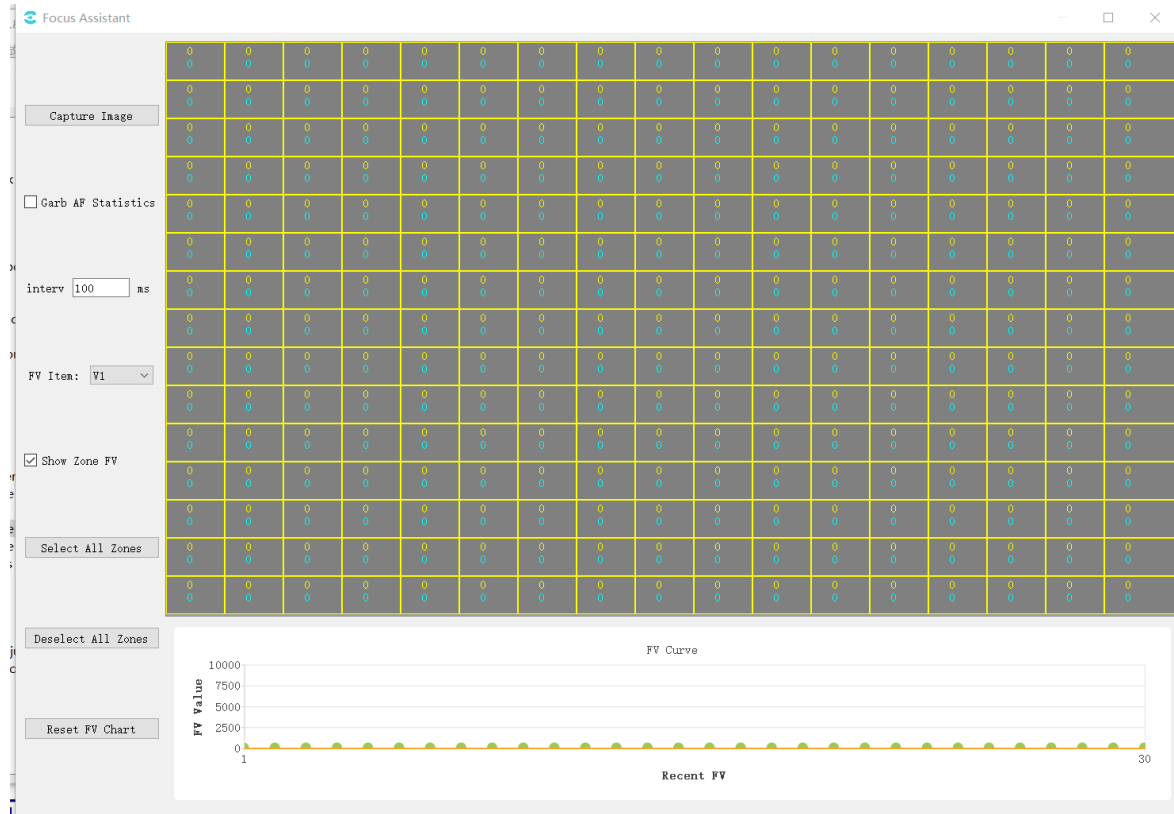


图 3.23: VLC 播放视频流

3.4.3 辅助对焦工具使用说明

3.4.3.1 工具界面

工具首页，选择工具栏 Extra Utilities->Focus Assistant，辅助对焦界面如下图



工具界面主要分三大部分：

- (1) 操作区：进行各种工具操作
- (2) 图像和区间显示区：显示抓拍到的图像，区间分布和各个区间的 FV 值（其中黄色值为区间当前的 FV 值，蓝色的值为区间历史 FV 最大值）
- (3) FV 变化曲线：显示全局 FV 值的变化曲线。其中横坐标为 1 的点为最近取到的 FV 值所表示的点。每次 FV 值刷新时，之前取到的值会右移。

3.4.3.2 利用辅助对焦工具进行调焦

使用辅助对焦工具调焦的步骤如下：

步骤 1. 将镜头焦距调整为最模糊状态

步骤 2. 确定当前设备的 AF 区间配置状态：勾选 Grab AF Statistics，并在图像显示区的区间数量变化时去除勾选。

步骤 3. 选择 FV 组别：在 FV Item 下拉框下选择 V1（垂直方向）和 H1 组（水平方向）。

步骤 4. 在图像和区间显示区域点击确定调焦时需要关注哪些区间。其中关注的区间左上角会显示一个绿色圆。可以点击工具左侧的 Select All Zones 选定所有区间，Deselect AllZones 取消所有区间选定，方便操作。

步骤 5. 再次勾选 Grab AF Statistics。此时图像显示区和下方的曲线图开始不断刷新。

步骤 6. 调整镜头焦距。此时 FV 值曲线最大值蓝线稳步升高。

步骤 7. 当发现 FV 值曲线上 1 位置的点开始出现平稳下降时，最大值蓝线的位置应该不再变化（可视为蓝线 FV 为理论最优对焦 FV）。此时反方向调焦至 1 位置上的点无限接近蓝线位置即可。

3.4.3.3 对焦辅助操作

如果用户在调焦过程中，需要确定点播图像上的区域属于哪个区间，可以点击 CaptureImage 来抓拍一帧图像。图像显示在图像区域后，即可进行确认。

工具上 Grab AF Statistics 勾选后，工具将以一个固定的时间间隔从板端获取 AF 统计信息。用户可以修改通过界面上的 Interval 值来改变这个间隔。改变范围为 100~1000 毫秒。

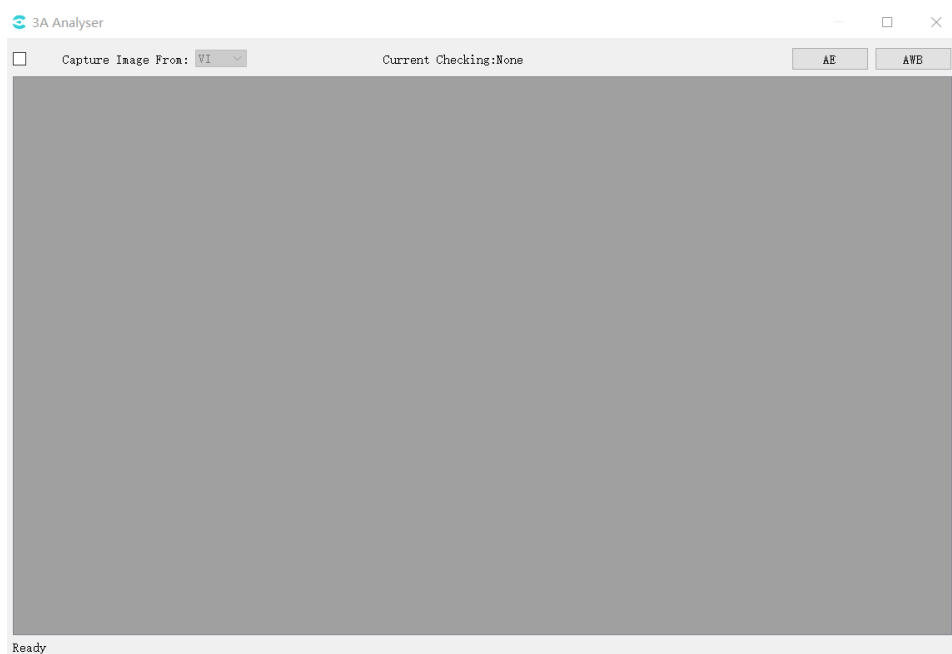
如果用户不需要查看分区间 FV 值的当前值和峰值，可以将左侧的 Show Zone FV 复选框去除勾选。去除勾选后再次勾选，区间 FV 值会再次显示。

如果用户改变了区间配置，此时的理论 FV 最大值会发生变化。建议用户此时点击“Reset FV Chart”按钮。点击后，右下方图表中的历史全局 FV 值将被清空，最大 FV 值的蓝线也将归零。

3.4.4 3A 分析工具

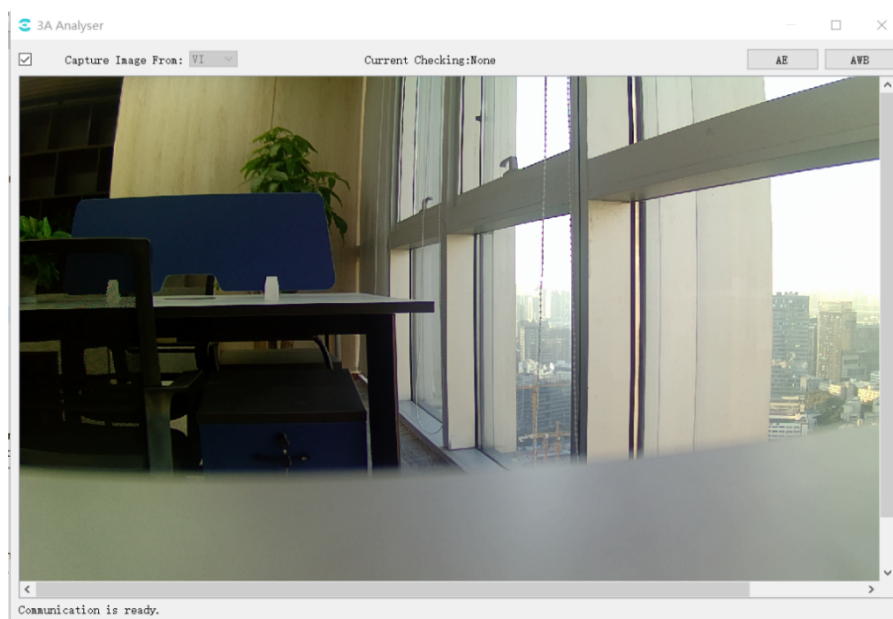
3.4.4.1 工具界面

工具首页，在工具栏选择 Extra Utilities -> 3A Analyser，3A 分析界面如下图：

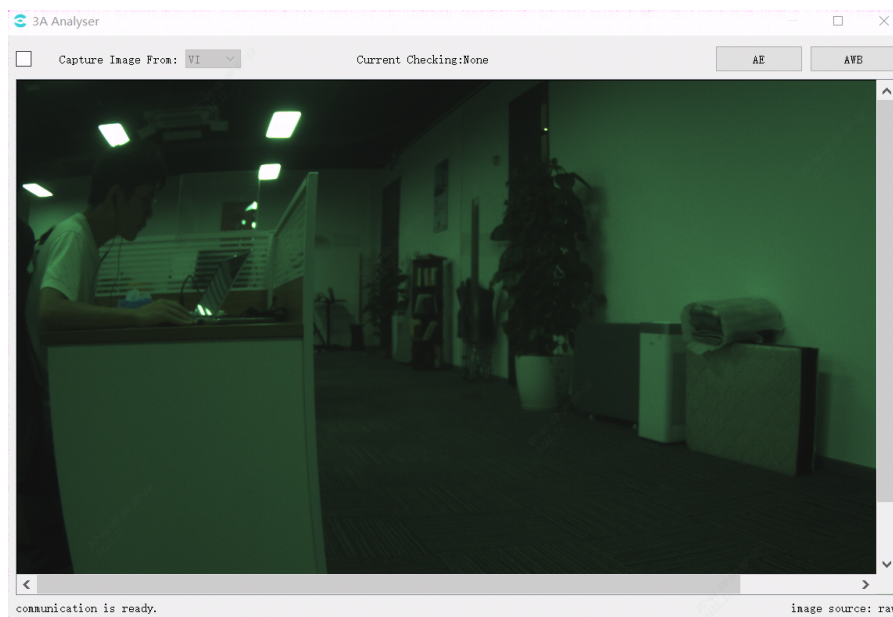


3.4.4.2 获取图像与统计数据

1. CviPQTool 与板端连接。
2. 在 3A 分析工具的主界面上，在 Capture Image from 之后的下拉框中，选择图像数据源（目前工具仅支持从 VI 上获取图像数据）。
3. 勾选 Capture Image from 之前的复选框。之后工具将自动从板端抓取数据，直到用户取消对 Capture Image 复选框的勾选。成功从板端抓取数据后，工具会在下方的灰色区域显示图像效果如下图所示：

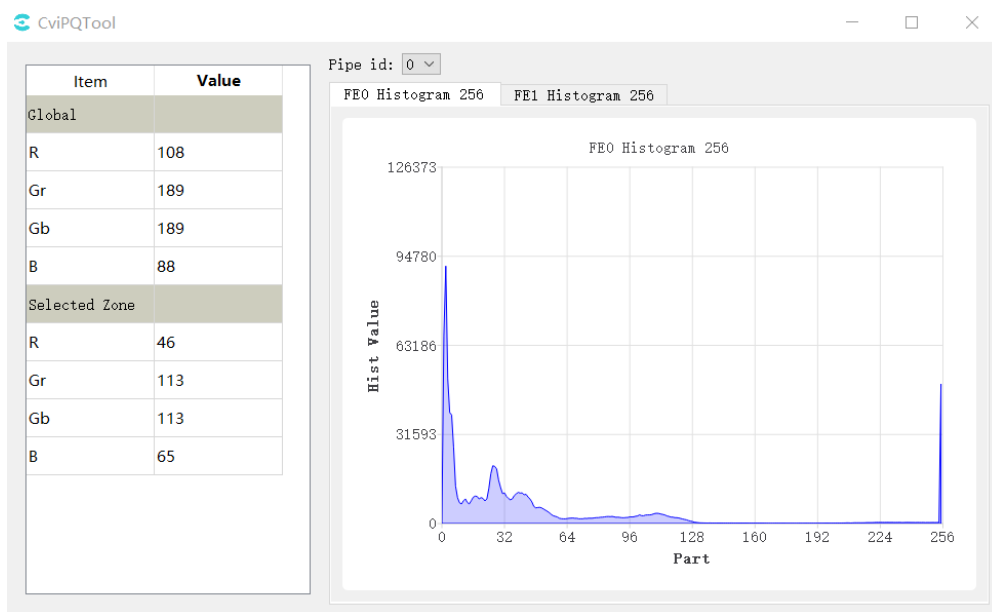


4. 但当板端无法获取到 YUV 图时，会使用 raw 图进行显示，raw 图显示的效果见下图：

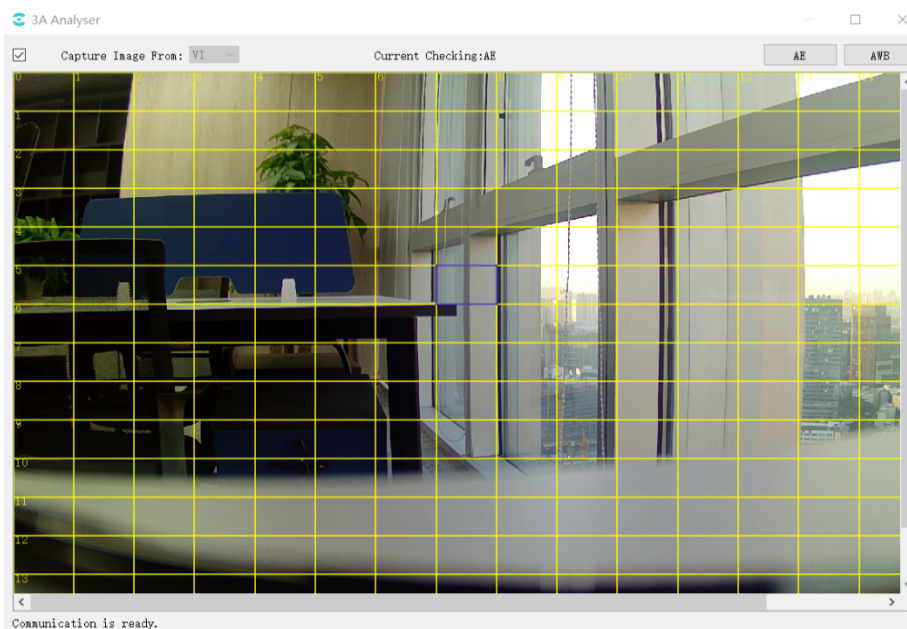


3.4.4.3 查看 AE 统计数据

点击 3A 分析工具界面右上角 AE 按钮，可以打开 AE 统计信息窗口如下图所示：



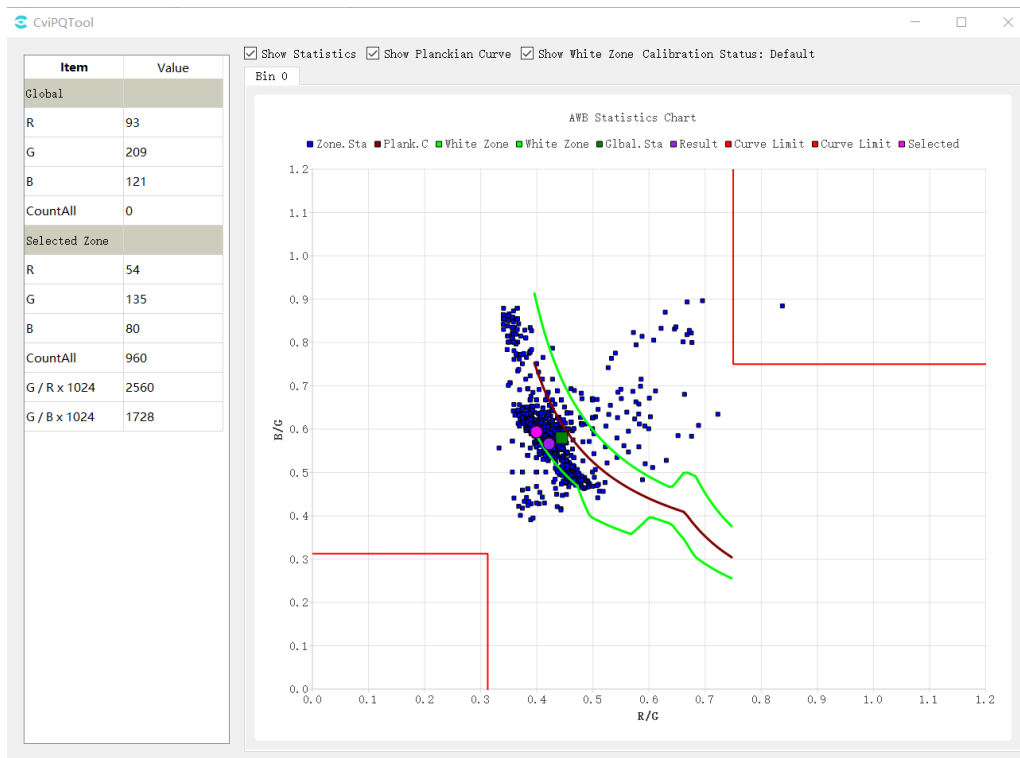
同时 3A 分析工具主界面图像区域也会被黄线分割成一个个区间，用户可以使用鼠标左键进行区间选择，如下图所示。



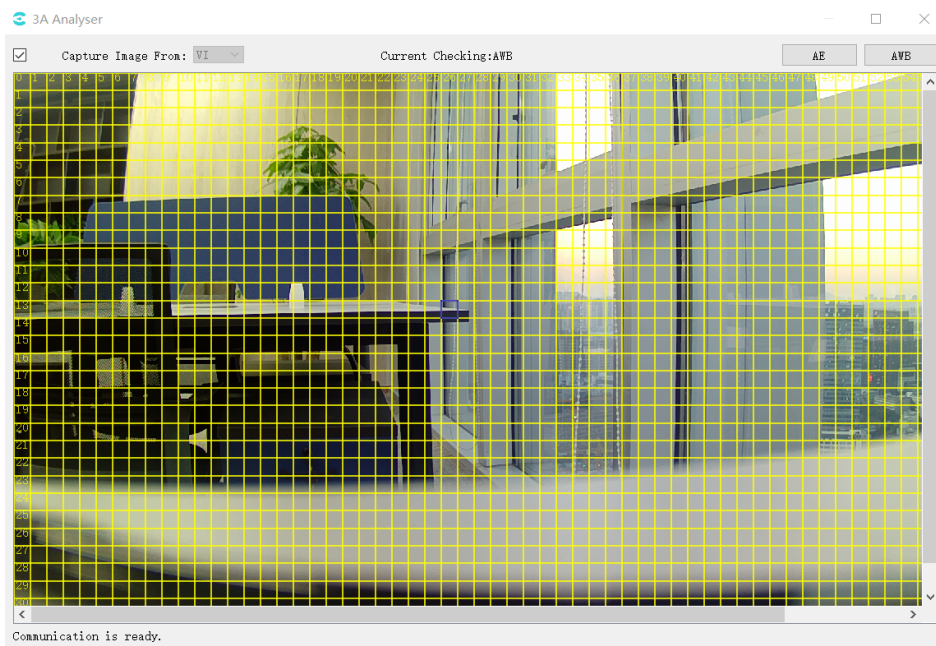
统计信息：可以查看全局（Global）和分区间（Selected Zone）的分量平均值。在 3A 分析工具主界面切换选区时，选择区间的统计信息数据会变化。

3.4.4.4 查看 AWB 统计数据

点击 3A 分析工具界面右上角 AWB 按钮，可以打开 AWB 统计信息窗口如下图所示：



同时，3A 分析工具的主界面图像区域也会被黄线分割成一个个区间，用户可以使用鼠标左键进行区间选择，如下图所示：



在统计图上显示的内容有：

红棕色色曲线：普朗克曲线 (plank.c)。

绿色点：全局 (global) 统计信息在普朗克坐标系上对应的点。

紫色点：AWB 计算结果在普朗克坐标系上对应的点。

蓝色点：分区间统计信息在普朗克坐标系上对应的点。

红色点：鼠标选择的图片分区 (selected zone) 在普朗克坐标系上对应的点。

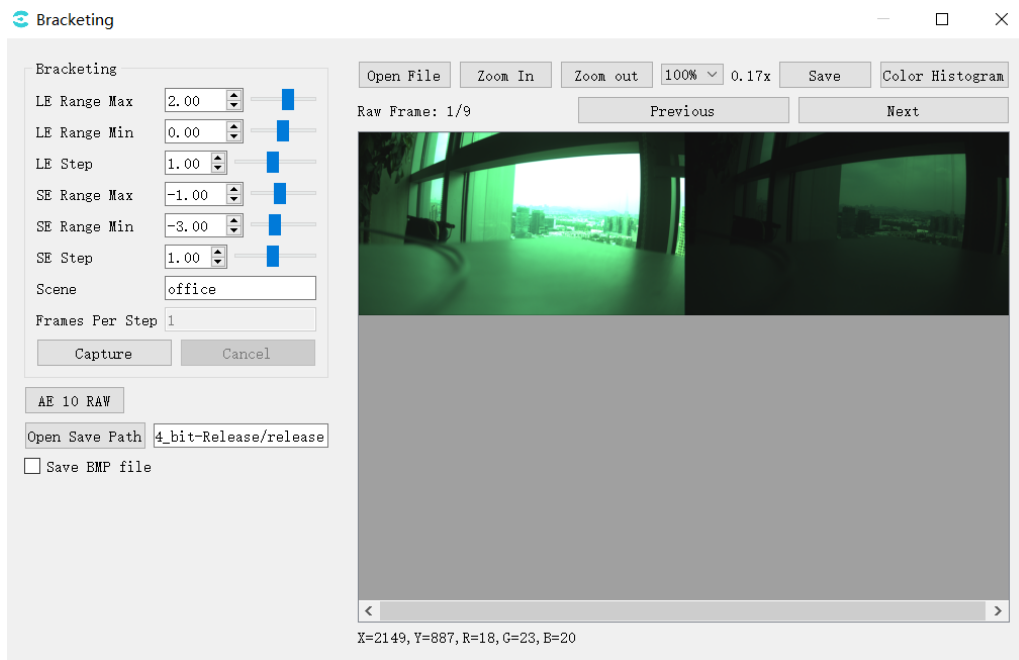
绿色曲线：基础白区区间。

右上和左下的红色直线围成的区域：右上和左下的红色直线围成的区域：分别为排除紫色和绿色干扰而从白区中剔除的区间。与 AWB 参数中的 CurveLLimit 和 CurveRLimit 相关

通过图像上方的 Show Statistics、ShowPlanckian Curve、ShowWhite Zone 可分别控制坐标图上统计信息、色温曲线和白区区间的显示与隐藏。

3.4.5 包围曝光

在工具栏选择 Extra Utilities -> Bracketing，包围曝光窗口如下图所示。



LE/SE Max: 设置长短曝最大值

LE/SE Min: 设置长短曝最小值

LE/SE Step: 步长

Capture: 抓取由 LE/SE 设置的不同曝光组合的 raw

AE 10 RAW: 抓取 10 张特定曝光的 raw

Save BMP file: 保存 BMP 图像，默认保存 JPG

Open Save Path: 文件保存目录

抓取张数计算：

$$\text{LeNum} = (\text{LeMax} - \text{LeMin}) / \text{LeStep} + 1;$$

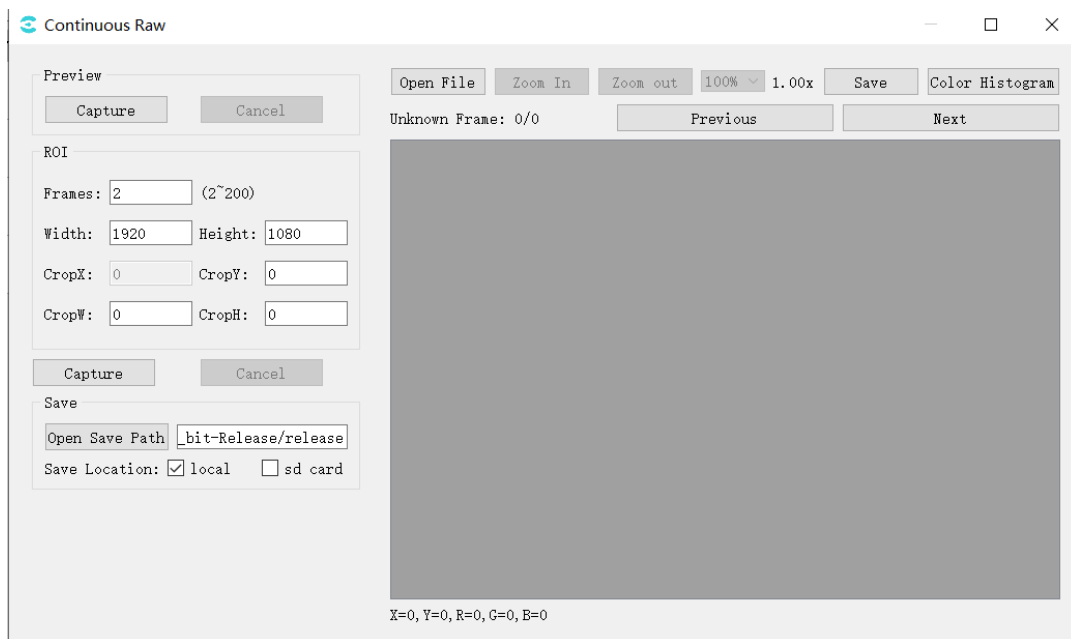
$$\text{SeNum} = (\text{SeMax} - \text{SeMin}) / \text{SeStep} + 1;$$

$$\text{wdr: TotalNum} = \text{LeNum} * \text{SeNum};$$

$$\text{linear: TotalNum} = \text{LeNum}$$

3.4.6 连续 raw 工具

在工具栏选择 Extra Utilities -> Continuous Raw，连续 raw 工具如下图所示。



Preview，预览当前画面

Capture：点击 capture，将抓取 yuv 并显示在右边窗口；

ROI，抓 raw 设置

Frames：抓取 raw 张数；

Width/Height：当前画面的宽高；

CropX/CropY/CropW/CropH：设置抓取 raw 的大小、位置；

Capture：开始抓取 raw；

Save，保存设置

Open Save Path：选择保存路径；

Save Location：选择保存位置，选择 local，raw 文件保存在 pc 端，选择 sd card，raw 文件保存在板端；

抓取文件如下图所示：

```

2560X1440 NV21 -bits=8 .jpg
2560X1440 GRBG Linear -color=2 -bits=12 -frame=1 -hdr=0 ISO=100 19700101100451-explInfo.txt
2560X1440 GRBG Linear -color=2 -bits=12 -frame=1 -hdr=0 ISO=100 19700101100451-awblog.txt
2560X1440 GRBG Linear -color=2 -bits=12 -frame=1 -hdr=0 ISO=100 19700101100451-awb.bin
2560X1440 GRBG Linear -color=2 -bits=12 -frame=1 -hdr=0 ISO=100 19700101100451-aelog.txt
2560X1440 GRBG Linear -color=2 -bits=12 -frame=1 -hdr=0 ISO=100 19700101100451 roi=2,0,0,1920,720.raw
2560X1440 GRBG Linear -color=2 -bits=12 -frame=1 -hdr=0 ISO=100 19700101100451.txt
2560X1440 GRBG Linear -color=2 -bits=12 -frame=1 -hdr=0 ISO=100 19700101100451.raw
2560X1440 GRBG Linear -color=2 -bits=12 -frame=1 -hdr=0 ISO=100 19700101100451.json

```

这里的 raw 图是经过压缩的，大小计算方法如下：

1. 全尺寸 raw

Size = (width * 6 / 8 + 15) / 16 * height * frame;

如果 Width > 2304, 压缩模式为 tile, Width 需要再加上 8;

2. roi raw

Size = (CropW * 6 / 8 + 15) / 16 * CropH * frame;

如果 Width > 2304, 且 CropW > 1536, 压缩模式为 tile, CropW 需要再加上 8;

3.5 参数详细说明

以下列出在工具界面上各模块参数对应的 SDK 的 API 函数参考的说明，如表所示。

表 3.2: TOP 设置对应 API

功能模块	对应 SDK API
Mode Handle	N/A
Bypass Setting	CVI_ISP_SetModuleControl CVI_ISP_GetModuleControl
Fmw State	CVI_ISP_SetFMWState CVI_ISP_GetFMWState
CTRL Param	CVI_ISP_SetCtrlParam CVI_ISP_GetCtrlParam
MOD Param	CVI_ISP_SetModParam CVI_ISP_GetModParam

表 3.3: PubAttr 设置对应 API

功能模块	对应 SDK API
PubAttr	CVI_ISP_SetPubAttr CVI_ISP_GetPubAttr

表 3.4: ISPInfo 设置对应 API

功能模块	对应 SDK API
ISPInfo	CVI_ISP_QueryInnerStateInfo

表 3.5: ExposureAttr 设置对应 API

功能模块	对应 SDK API
ExposureAttr	CVI_ISP_SetExposureAttr CVI_ISP_GetExposureAttr

表 3.6: WDRExposureAttr 设置对应 API

功能模块	对应 SDK API
WDRExposureAttr	CVI_ISP_SetWDRExposureAttr CVI_ISP_GetWDRExposureAttr

表 3.7: ExposureInfo 设置对应 API

功能模块	对应 SDK API
ExposureInfo	CVI_ISP_QueryExposureInfo

表 3.8: AERoute 设置对应 API

功能模块	对应 SDK API
AERoute	CVI_ISP_SetAERouteAttr CVI_ISP_GetAERouteAttr CVI_ISP_GetAERouteAttrEx CVI_ISP_SetAERouteAttrEx

表 3.9: WBAttr 设置对应 API

功能模块	对应 SDK API
White Balance	CVI_ISP_SetWBAttr CVI_ISP_GetWBAttr
AWBAttr	CVI_ISP_SetWBAttr CVI_ISP_GetWBAttr
MWBAttr	CVI_ISP_SetWBAttr CVI_ISP_GetWBAttr
AWB Calibration Data	CVI_ISP_GetWBCalibration CVI_ISP_SetWBCalibration
AWBAttrEx	CVI_ISP_GetAWBAttrEx CVI_ISP_SetAWBAttrEx

表 3.10: WBInfo 设置对应 API

功能模块	对应 SDK API
WBInfo	CVI_ISP_QueryWBInfo

表 3.11: CCM 设置对应 API

功能模块	对应 SDK API
CCM	CVI_ISP_SetCCMAAttr CVI_ISP_GetCCMAAttr
ColorToneAttr	CVI_ISP_GetColorToneAttr CVI_ISP_SetColorToneAttr

表 3.12: Demosaic 设置对应 API

功能模块	对应 SDK API
Demosaic	CVI_ISP_SetDemosaicAttr CVI_ISP_GetDemosaicAttr CVI_ISP_SetDemosaicDemoireAttr CVI_ISP_GetDemosaicDemoireAttr

表 3.13: 3DNR 设置对应 API

功能模块	对应 SDK API
3DNR	CVI_ISP_SetTNRAAttr CVI_ISP_GetTNRAAttr CVI_ISP_SetTNRMAAttr CVI_ISP_GetTNRMAAttr CVI_ISP_SetTNRPAAttr CVI_ISP_GetTNRPAAttr CVI_ISP_SetTNRNRAAttr CVI_ISP_GetTNRNRAAttr

表 3.14: DRC 设置对应 API

功能模块	对应 SDK API
DRC	CVI_ISP_SetDRCAAttr CVI_ISP_GetDRCAAttr

表 3.15: Crosstalk 设置对应 API

功能模块	对应 SDK API
Crosstalk	CVI_ISP_GetCrosstalkAttr CVI_ISP_SetCrosstalkAttr

表 3.16: Shading 设置对应 API

功能模块	对应 SDK API
Shading Attr	CVI_ISP_SetMeshShadingAttr CVI_ISP_GetMeshShadingAttr CVI_ISP_SetRadialShadingAttr CVI_ISP_GetRadialShadingAttr
Shading Lut Attr	CVI_ISP_SetMeshShadingGainLutAttr CVI_ISP_GetMeshShadingGainLutAttr CVI_ISP_SetRadialShadingGainLutAttr CVI_ISP_GetRadialShadingGainLutAttr

表 3.17: DPC 设置对应 API

功能模块	对应 SDK API
Dynamic Attribute	CVI_ISP_GetDPDynamicAttr CVI_ISP_SetDPDynamicAttr

表 3.18: CNR 设置对应 API

功能模块	对应 SDK API
CNR Attr	CVI_ISP_SetCNRAttr CVI_ISP_GetCNRAttr
CNR FILTER Attr	CVI_ISP_SetCNRFilterAttr CVI_ISP_GetCNRFilterAttr

表 3.19: DCI 设置对应 API

功能模块	对应 SDK API
DCI	CVI_ISP_SetDCIAttr CVI_ISP_GetDCIAttr
DCI AUTO GAMMA ATTR	CVI_ISP_SetDciAutoGammaAttr CVI_ISP_GetDciAutoGammaAttr

表 3.20: Gamma 设置对应 API

功能模块	对应 SDK API
Gamma	CVI_ISP_SetGammaAttr CVI_ISP_GetGammaAttr

表 3.21: FSHDR 设置对应 API

功能模块	对应 SDK API
FSHDR	CVI_ISP_SetFSHDRAttr CVI_ISP_GetFSHDRAttr

表 3.22: LUT 3D 设置对应 API

功能模块	对应 SDK API
LUT 3D	CVI_ISP_SetClutAttr CVI_ISP_GetClutAttr CVI_ISP_SetClutHslAttr CVI_ISP_GetClutHslAttr

表 3.23: Y Sharpen 设置对应 API

功能模块	对应 SDK API
Sharpen	CVI_ISP_GetSharpenAttr CVI_ISP_SetSharpenAttr

表 3.24: BLC 设置对应 API

功能模块	对应 SDK API
Black Level	CVI_ISP_GetBlackLevelAttr CVI_ISP_SetBlackLevelAttr

表 3.25: BNR 设置对应 API

功能模块	对应 SDK API
BNR	CVI_ISP_SetBNRAAttr CVI_ISP_GetBNRAAttr
Filter	CVI_ISP_SetBNRFilterAttr CVI_ISP_GetBNRFilterAttr

表 3.26: Saturation 设置对应 API

功能模块	对应 SDK API
Saturation	CVI_ISP_GetSaturationAttr CVI_ISP_SetSaturationAttr

表 3.27: Statistics Config 设置对应 API

功能模块	对应 SDK API
AE/AF/AWB Config	CVI_ISP_GetStatisticsConfig CVI_ISP_SetStatisticsConfig

表 3.28: LDC 设置对应 API

功能模块	对应 SDK API
VI LDC	CVI_VI_GetChnLDCAttr CVI_VI_SetChnLDCAttr
VPSS LDC	CVI_VPSS_GetChnLDCAttr CVI_VPSS_SetChnLDCAttr

表 3.29: Mono 设置对应 API

功能模块	对应 SDK API
Mono	CVI_ISP_GetMonoAttr CVI_ISP_SetMonoAttr

表 3.30: Y Contrast 设置对应 API

功能模块	对应 SDK API
YContrast Attr	CVI_ISP_GetYContrastAttr CVI_ISP_SetYContrastAttr

表 3.31: Vpss Adjustment 设置对应 API

功能模块	对应 SDK API
VPSS Adjustment	CVI_VPSS_GetGrpProcAmp CVI_VPSS_SetGrpProcAmp

表 3.32: Noise Profile 设置对应 API

功能模块	对应 SDK API
NoiseProfile	CVI_ISP_GetNoiseProfileAttr CVI_ISP_SetNoiseProfileAttr

表 3.33: PFR 设置对应 API

功能模块	对应 SDK API
PFR	CVI_ISP_SetPFRAttr CVI_ISP_GetPFRAttr

4 工具应用参考

4.1 工具的参数如何导入导出？

4.1.1 使用工具导入导出参数

CviPQ Tool 支持在 PC 端的 json 格式配置文件导入和导出，并且支持将工具的参数固化到板端。如果用户想要导出或导入 CviPQ Tool 的参数文件，可参考章节2.4.3.1. 和2.4.3.2. 的说明进行操作。

如果用户想要将参数导入到板端或者从板端导出到 PC 端进行备份，请使用二进制实用工具（Bin Utility 工具）进行操作。具体的操作说明如下：

- 导入导出板端参数的 BIN 文件

在已连接到板端的情况下，点击菜单栏中的 Bin Utility，即可打开二进制实用工具窗口，如图 所示：

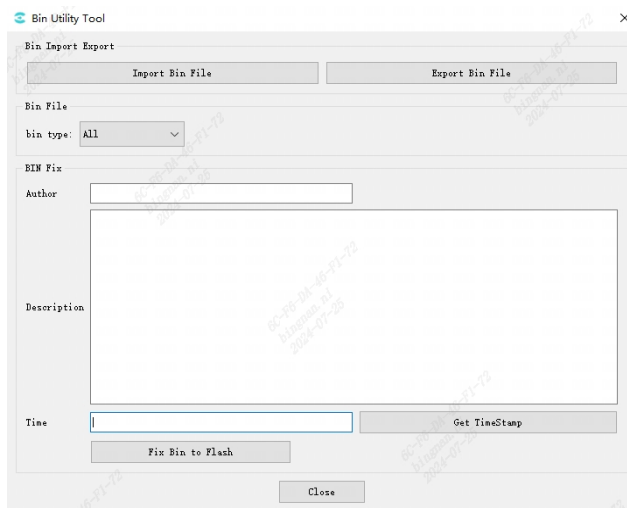


图 4.1: 二进制实用工具

在 Bin Import Export 群组里有两个按钮，分别点击按钮可完成如下操作：

- 导入参数 bin 文件到板端：点击“Import Bin File”在弹出的打开文件对话框中选择需要导入的正确参数 bin 文件，当工具完成发送文件到板端时，参数会立即生效。

- 导出板端参数 bin 文件：点击 bin type 下拉框选择导出类型，然后点击“Export Bin File”在弹出的保存文件对话框中选择一个保存路径，工具会将板端上选中类型的参数保存在指定的路径。
- 将参数配置固化到 Flash

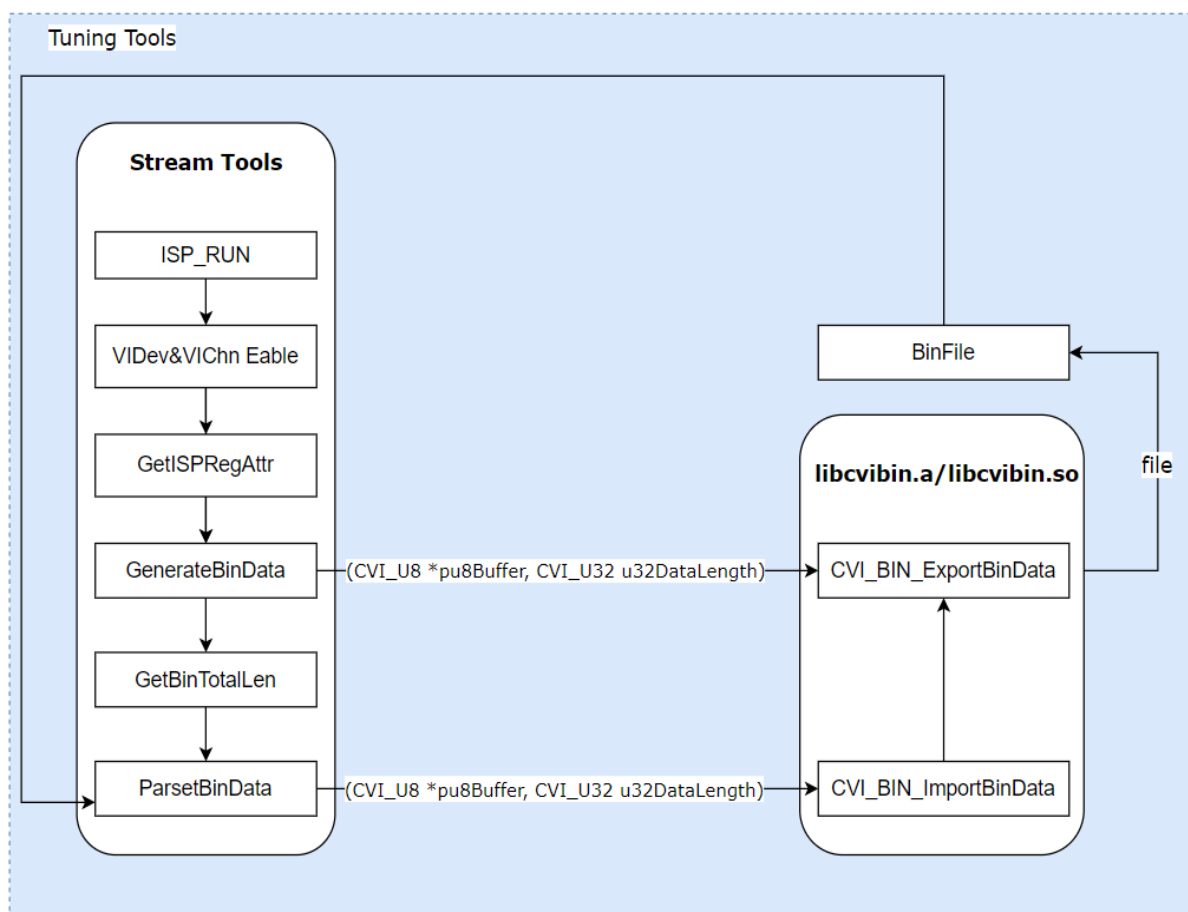
在 Bin Fix 群组内填写作者、描述信息和时间信息后，可点击“Fix Bin to Flash”向板端发送指令，将当前板端的参数信息写入到 Flash 中。

4.1.2 使用库导入导出图像质量参数

板端工具提供导入导出参数库文件，以下两种方式可根据自己的需要进行选择。**注意：以下接口必须在调用 CVI_ISP_Init 接口之后才能调用。**

1. 获取和保存所有模块 bin 数据 (ISP x、VPSS、VO 等):
 - CVI_BIN_GetBinTotalLen: 获取 bin 数据的总长度;
 - CVI_BIN_ExportBinData: 导出所有模块 bin 数据;
 - CVI_BIN_ImportBinData: 导入 bin 数据，会根据 bin 中的模块数据和当前的所有模块进行匹配，将所有符合条件的数据导入;
2. 获取和保存单个模块 bin 数据 (ISP x、VPSS、VO 等):
 - CVI_BIN_GetSingleISPBINLen: 获取某个模块的 bin 数据的长度;
 - CVI_BIN_ExportSingleISPBINData: 导出单个模块 bin 数据;
 - CVI_BIN_SaveParamToBin: 导出所有模块 bin 数据;
 - CVI_BIN_LoadParamFromBin: 导入单个模块 bin 数据;
 - CVI_BIN_LoadParamFromBinEx: 导入单个模块 bin(含输入 buf 长度) 数据;

可参考推荐使用流程，如下图所示。



其他相关接口 API:

- **CVI_BIN_SetBinName**: 设置 PQBin 存放的路径和文件名;
- **CVI_BIN_GetBinName**: 获取 PQBin 存放的路径和文件名;
- **CVI_BIN_GetBinExtraAttr**: 获取 bin 头数据信息;
- **CVI_ISP_BIN_SetBypassParams**: 设置 bin 加载时参数的失效与否, 默认都生效;
- **CVI_ISP_BIN_GetBypassParams**: 获取 bin 加载时参数的生效信息;

4.1.3 API 参考

4.1.3.1 CVI_BIN_GetBinTotalLen

【描述】

获取 bin 数据的总长度。

【语法】

```
CVI_U32 CVI_BIN_GetBinTotalLen(void);
```

【参数】

参数名称	描述	输入/输出
无	无	无

【返回值】

返回值	描述
CVI_U32	bin 文件的总长度

【需求】

- 头文件: `cvi_bin.h`
- 库文件: `libcvi_bin.so`, `libcvi_bin_isp.so`

【注意】

此函数必须在调用 `CVI_BIN_ExportBinData` 接口前调用。

【举例】

```
CVI_U32 u32BinLen = 0;  
u32BinLen = CVI_BIN_GetBinTotalLen();
```

【相关主题】

- `CVI_BIN_ExportBinData`

4.1.3.2 CVI_BIN_ExportBinData

【描述】

导出所有模块 bin 数据。

【语法】

```
CVI_S32 CVI_BIN_ExportBinData(CVI_U8*pu8Buffer, CVI_U32 u32DataLength);
```

【参数】

参数名称	描述	输入/输出
<code>pu8Buffer</code>	保存 pqbin 资料的 buffer	输入
<code>u32DataLength</code>	pqbin 资料的总长度	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值请参见 错误码 1 。

【需求】

- 头文件: `cvi_bin.h`
- 库文件: `libcvi_bin.so`, `libcvi_bin_isp.so`

【注意】

调用此函数必须先调用CVI_BIN_GetBinTotalLen 函数，获取数据的大小，否则可能出现内存问题。

【举例】

```
CVI_S32 ret = CVI_SUCCESS;

CVI_BIN_HEADER header;

CVI_U32 u32BinLen = 0, u32TempLen = 0;

CVI_U8 *pBuffer;

memset(&header, 0, sizeof(CVI_BIN_HEADER));

u32BinLen = CVI_BIN_GetBinTotalLen();

pBuffer = (CVI_U8 *)malloc(u32BinLen);

if (pBuffer == NULL) {

ISP_DAEMON_DEBUG(LOG_ALERT, "malloc err!\\n");

return CVI_FAILURE;

}

header.extraInfo = *binExtra;

memcpy(pBuffer, &header, sizeof(CVI_BIN_HEADER));

ret = CVI_BIN_ExportBinData(pBuffer, u32BinLen);

if (ret != CVI_SUCCESS) {

ISP_DAEMON_DEBUG_EX(LOG_ALERT, "CVI_BIN_ExportBinData err(%#x)!\\n", ret);

} else {

u32TempLen = fwrite(pBuffer, 1, u32BinLen, fp);

if (u32TempLen != u32BinLen) {

ISP_DAEMON_DEBUG(LOG_ALERT, "writeIspRegToBin fail\\n");

ret = CVI_FAILURE;

}

}

if (pBuffer != NULL) {

free(pBuffer);
```

(下页继续)

(续上页)

```
}  
  
ISP_DAEMON_UNUSED(numDevice);  
  
return ret;
```

【相关主题】

无

4.1.3.3 CVI_BIN_ImportBinData**【描述】**

导入所有模块 bin 数据，会根据 bin 中的模块数据和当前所有的模块进行匹配，将所有符合条件的数据导入。

【语法】

```
CVI_S32 CVI_BIN_ImportBinData(CVI_U8*pu8Buffer, CVI_U32 u32DataLength);
```

【参数】

参数名称	描述	输入/输出
pu8Buffer	Bin 的内存空间	输入
u32DataLength	Bin 大小	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值请参见 错误码 2

【需求】

- 头文件：cvi_bin.h
- 库文件：libcvi_bin.so libcvi_bin_isp.so

【注意】

无

【举例】

```
CVI_S32 ret = CVI_SUCCESS;  
  
FILE *fp = NULL;  
  
CVI_U8 *buf = NULL;  
  
CVI_CHAR binName[BIN_FILE_LENGTH] = { 0 };
```

(下页继续)

(续上页)

```
CVI_U32 u32TempLen = 0, u32FileSize = 0;

ret = CVI_BIN_GetBinName(binName);

if (ret != CVI_SUCCESS) {
    CVI_TRACE_SYS(CVI_DBG_WARN, "GetBinName(%s) fail\\n", binName);
}

fp = fopen((const CVI_CHAR *)binName, "rb");

if (fp == NULL) {
    CVI_TRACE_SYS(CVI_DBG_WARN, "Can't find bin(%s)\\n", binName);
    ret = CVI_FAILURE;
    goto ERROR_HANDLER;
} else {
    CVI_TRACE_SYS(CVI_DBG_WARN, "Bin exist (%s)\\n", binName);
}

getFileSize(fp, &u32FileSize);

buf = (CVI_U8 *)malloc(u32FileSize);

if (buf == NULL) {
    ret = CVI_FAILURE;
    CVI_TRACE_SYS(CVI_DBG_WARN, "Allocate memory fail\\n");
    goto ERROR_HANDLER;
}

u32TempLen = fread(buf, u32FileSize, 1, fp);

if (u32TempLen <= 0) {
    CVI_TRACE_SYS(CVI_DBG_WARN, "read data to buff fail!\\n");
    ret = CVI_FAILURE;
    goto ERROR_HANDLER;
}

ret = CVI_BIN_ImportBinData(buf, (CVI_U32)u32FileSize);
```

(下页继续)

(续上页)

```
if (ret != CVI_SUCCESS) {  
    CVI_TRACE_SYS(CVI_DBG_WARN, "CVI_BIN_ImportBinData error! value:(0x%x)\\n", ret);  
    goto ERROR_HANDLER;  
}  
  
ERROR_HANDLER:  
  
if (fp != NULL) {  
    fclose(fp);  
}  
  
if (buf != NULL) {  
    free(buf);  
}  
  
return ret;
```

【相关主题】

无

4.1.3.4 CVI_BIN_GetSingleISPBinLen**【描述】**

获取某个模块的 bin 数据的长度。

【语法】

CVI_S32 CVI_BIN_GetSingleISPBinLen(enum CVI_BIN_SECTION_ID id);

【参数】

参数名称	描述	输入/输出
id	模块 ID	输入

【返回值】

返回值	描述
CVI_U32	某个模块的 bin 文件长度

【需求】

- 头文件: cvi_bin.h
- 库文件: libcvi_bin.so, libcvi_bin_isp.so

【注意】

此函数必须在调用 `CVI_BIN_ExportSingleISPBinData` 接口前调用。

【举例】

```
CVI_U32 u32BinLen = 0;
enum CVI_BIN_SECTION_ID id = CVI_BIN_ID_ISP0;
u32BinLen = CVI_BIN_GetBinTotalLen(id);
```

【相关主题】

- `CVI_BIN_ExportSingleISPBinData`

4.1.3.5 CVI_BIN_ExportSingleISPBinData**【描述】**

导出单个模块的 bin 数据。

【语法】

```
CVI_BIN_ExportSingleISPBinData(enum CVI_BIN_SECTION_ID id, CVI_U8*pu8Buffer,
CVI_U32 u32DataLength);
```

【参数】

参数名称	描述	输入/输出
id	模块 ID	输入
pu8Buffer	保存 pqbin 资料的 buffer	输入
u32DataLength	pqbin 资料的总长度	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值请参见 错误码 1 。

【需求】

- 头文件: `cvi_bin.h`
- 库文件: `libcvi_bin.so`, `libcvi_bin_isp.so`

【注意】

调用此函数必须先调用 `CVI_BIN_GetSingleISPBinLen` 函数, 获取数据的大小, 否则可能出现内存问题。

【举例】

```
CVI_S32 ret = CVI_SUCCESS;
CVI_BIN_HEADER header;
```

(下页继续)

(续上页)

```
CVI_U32 u32BinLen = 0, u32TempLen = 0;

CVI_U8 *pBuffer;

enum CVI_BIN_SECTION_ID id = CVI_BIN_ID_ISP0;

memset(&header, 0, sizeof(CVI_BIN_HEADER));

u32BinLen = CVI_BIN_GetSingleISPBInLen(id);

pBuffer = (CVI_U8 *)malloc(u32BinLen);

if (pBuffer == NULL) {

ISP_DAEMON_DEBUG(LOG_ALERT, "malloc err!\n");

return CVI_FAILURE;

}

header.extraInfo = *binExtra;

memcpy(pBuffer, &header, sizeof(CVI_BIN_HEADER));

ret = CVI_BIN_ExportSingleISPBInData(id, pBuffer, u32BinLen);

if (ret != CVI_SUCCESS) {

ISP_DAEMON_DEBUG_EX(LOG_ALERT, "CVI_BIN_ExportBinData err(%#x)!\n", ret);

} else {

u32TempLen = fwrite(pBuffer, 1, u32BinLen, fp);

if (u32TempLen != u32BinLen) {

ISP_DAEMON_DEBUG(LOG_ALERT, "writeIspRegToBin fail!\n");

ret = CVI_FAILURE;

}

}

if (pBuffer != NULL) {

free(pBuffer);

}

ISP_DAEMON_UNUSED(numDevice);

return ret;
```

【相关主题】

无

4.1.3.6 CVI_BIN_SaveParamToBin**【描述】**

导出单个模块的 bin 数据

【语法】

```
CVI_S32 CVI_BIN_SaveParamToBin(FILE*fp, CVI_BIN_EXTRA_S*extraInfo);
```

【参数】

参数名称	描述	输入/输出
fp	PQBin 文件指针	输入
extraInfo	PQBin 信息和描述	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值请参见 错误码 1

【需求】

- 头文件: cvi_bin.h
- 库文件: libcvi_bin.so, libcvi_bin_isp.so

【注意】

无

【举例】

```
CVI_BIN_EXTRA_S BinExtra = {  
    "User",  
    "test-default",  
    "2021-04-19",  
};  
  
FILE *fd = fopen(TEST_BIN, "wb");  
  
if (fd == NULL) {  
    ISP_DEBUG(LOG_ERR, "Open file failed\\n");  
}
```

(下页继续)

(续上页)

```
CVI_BIN_SaveParamToBin(fd, &BinExtra);  
fclose(fd);
```

【相关主题】

无

4.1.3.7 CVI_BIN_LoadParamFromBin**【描述】**

导入单个模块 bin 数据。

【语法】

```
CVI_S32 CVI_BIN_LoadParamFromBin(enum CVI_BIN_SECTION_ID id, CVI_U8*buf);
```

【参数】

参数名称	描述	输入/输出
id	模块 ID	输入
buf	PQBin 的数据内容	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 2

【需求】

- 头文件: cvi_bin.h
- 库文件: libcvi_bin.so, libcvi_bin_isp.so

【注意】

须先将 PQBin 从数据读到缓存内存空间, 再呼叫此函数式。此 API 的功能之后会由 CVI_BIN_LoadParamFromBinEx 所取代, 建议改用。

【举例】

```
CVI_S32 result = CVI_FAILURE;  
  
CVI_U32 size = 0;  
  
FILE *fp;  
  
ISP_DEBUG(LOG_DEBUG, "try load default value from bin %s\\n", TEST_BIN);  
  
fp = fopen(TEST_BIN, "rb");
```

(下页继续)

(续上页)

```
if (fp == NULL) {
ISP_DEBUG(LOG_WARNING, "Cant find bin(%s)\n", TEST_BIN);
} else {
ISP_DEBUG(LOG_INFO, "Bin exist (%s)\n", TEST_BIN);
}

fseek(fp, 0L, SEEK_END);

size = ftell(fp);

rewind(fp);

if (size > 0) {
// allocate buffer

CVI_U8 *binBuffer = malloc(size);

fread(binBuffer, size, 1, fp);

for (CVI_U32 idx = CVI_BIN_ID_MIN ; idx < CVI_BIN_ID_MAX ; idx++) {
result = CVI_BIN_LoadParamFromBin((enum CVI_BIN_SECTION_ID)idx, binBuffer);
}

// free buffer

free(binBuffer);

if (result == CVI_SUCCESS) {
ISP_DEBUG(LOG_DEBUG, "load default value from bin %s\n", TEST_BIN);
}
}
```

【相关主题】[CVI_BIN_LoadParamFromBinEx](#)

4.1.3.8 CVI_BIN_LoadParamFromBinEx

【描述】

同CVI_BIN_LoadParamFromBin。

【语法】

```
CVI_S32 CVI_BIN_LoadParamFromBin(enum CVI_BIN_SECTION_ID id, CVI_U8*buf, CVI_U32 u32DataLength);
```

【参数】

参数名称	描述	输入/输出
id	模块 ID	输入
buf	PQBin 的数据内容	输入
u32DataLength	buf 的大小	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 2

【需求】

- 头文件: cvi_bin.h
- 库文件: libcvi_bin.so, libcvi_bin_isp.so

【注意】

同CVI_BIN_LoadParamFromBin。

【举例】

请参考CVI_BIN_LoadParamFromBin，注意的是：需将 buf 的大小传入。

4.1.3.9 CVI_BIN_SetBinName

【描述】

设置 PQBin 存放的路径和文件名。

【语法】

```
CVI_S32 CVI_BIN_SetBinName(WDR_MODE_E wdrMode, const CVI_CHAR *binName);
```

【参数】

参数名称	描述	输入/输出
wdrMode	Sensor wdr mode	输入
binName	设定和 wdr mode 对应 pqbin 的路径和文件名 例 如” /mnt/data/cvi_sdr_bin”	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码。

【需求】

- 头文件: `cvi_bin.h`
- 库文件: `libcvi_bin.so`, `libcvi_bin_isp.so`

【注意】

无

【举例】

无

【相关主题】

[CVI_BIN_GetBinName](#)

4.1.3.10 CVI_BIN_GetBinName

【描述】

获取 PQBin 存放的路径和文件名。

【语法】

```
CVI_S32 CVI_BIN_GetBinName(CVI_CHAR*binName);
```

【参数】

参数名称	描述	输入/输出
<code>binName</code>	pqbin 路径和文件名	输出

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码。

【需求】

- 头文件: `cvi_bin.h`
- 库文件: `libcvi_bin.so`, `libcvi_bin_isp.so`

【注意】

无

【举例】

无

【相关主题】

CVI_BIN_SetBinName

4.1.3.11 CVI_BIN_GetBinExtraAttr

【描述】

获取 bin 头数据信息。

【语法】

```
CVI_S32 CVI_BIN_GetBinExtraAttr(FILE*fp, CVI_BIN_EXTRA_S*extraInfo)
```

【参数】

参数名称	描述	输入/输出
fp	PQBin 文件指针	输入
extraInfo	bin 头信息的内存空间	输入

【返回值】

返回值	描述
0	成功。

【需求】

- 头文件: cvi_bin.h
- 库文件: libcvi_bin.so, libcvi_bin_isp.so

【注意】

无

【举例】

```
CVI_BIN_EXTRA_S BinExtraAttr = {};  
  
FILE *fd = fopen("/mnt/data/bin/cvi_sdr_bin", "rb");  
  
CVI_BIN_GetBinExtraAttr(fd, &BinExtraAttr);
```

【相关主题】

无

4.1.3.12 CVI_ISP_BIN_SetBypassParams

【描述】

设置 bin 加载时参数的失效与否，默认都生效。

【语法】

```
CVI_S32 CVI_ISP_BIN_SetBypassParams(enum CVI_BIN_SECTION_ID id,
ISP_BIN_BYPASS_U * ispBinBypass)
```

【参数】

参数名称	描述	输入/输出
id	模块 id	输入
ispBinBypass	bin 参数生效信息	输入

【返回值】

返回值	描述
0	成功
-1	失败
0xCB000013	id 错误

【需求】

- 头文件: cvi_bin.h
- 库文件: libcvi_bin.so, libcvi_bin_isp.so

【注意】

无

【举例】

```
CVI_S32 ret = CVI_SUCCESS;

ISP_BIN_BYPASS_U ispBinBypass = {0};

ispBinBypass.bitBypassFrameRate = CVI_TRUE;

ret = CVI_ISP_BIN_SetBypassParams(CVI_BIN_ID_ISP0, &ispBinBypass);
```

【相关主题】

无

4.1.3.13 CVI_ISP_BIN_GetBypassParams

【描述】

设置 bin 加载时参数的失效与否，默认都生效。

【语法】

```
CVI_S32 CVI_ISP_BIN_GetBypassParams(enum CVI_BIN_SECTION_ID id,
ISP_BIN_BYPASS_U * ispBinBypass)
```

【参数】

参数名称	描述	输入/输出
id	模块 id	输入
ispBinBypass	bin 参数生效信息	输出

【返回值】

返回值	描述
0	成功
-1	失败
0xCB000013	id 错误

【需求】

- 头文件: cvi_bin.h
- 库文件: libcvi_bin.so, libcvi_bin_isp.so

【注意】

无

【举例】

```
CVI_S32 ret = CVI_SUCCESS;

ISP_BIN_BYPASS_U ispBinBypass = {0};

ret = CVI_ISP_BIN_GetBypassParams(CVI_BIN_ID_ISP0, &ispBinBypass);
```

【相关主题】

无

4.1.3.14 错误码 1

接口返回值	含义
0xCB000001	入参指针为空
0xCB000003	未分配到内存空间
0xCB00000B	json 压缩失败
0xCB00000D	输入的 buffer 空间不足
0xCB00000F	更新参数到 PQbin 文件失败
0xCB000012	创建 json 句柄失败
0xCB000013	输入的模块 id 无效
0xCB000014	从文件中读参数失败

4.1.3.15 错误码 2

接口返回值	含义
0xCB000001	入参指针为空
0xCB000003	未分配到内存空间
0xCB000006	输入 buf 的大小为 0
0xCB000008	PQbin 中的数据有异常
0xCB00000C	json 解压缩失败
0xCB00000F	更新参数到 PQbin 文件失败
0xCB000010	无法找到 bin 文件
0xCB000011	当前 PQbin 文件中 json 参数是无效的
0xCB000012	创建 json 句柄失败
0xCB000013	输入的模块 id 无效
0xCB000014	从文件中读参数失败
0xCB000015	当前所用的 PQbin 文件是无效的
0xCB000016	Sensor 数目超出 PQbin 文件指定的数目

5 FAQ

5.1 为什么工具打开调试数据文件时会提示版本不匹配?

当工具打开调试数据文件时会去检查文件内的版本信息，版本信息范例如下图：

```
"JSON Info": [  
  {  
    "key": "SoC",  
    "value": "CV183X"  
  },  
  {  
    "key": "CviPQ Tool Version",  
    "value": "v1.8.0.0"  
  },  
  {  
    "key": "Parameters Version",  
    "value": "cv183x v1.0.0"  
  },  
  {  
    "key": "DateTime",  
    "value": "20210421214211"  
  }  
],
```

图 5.1: 调试数据文件内的版本信息

如果调试数据文件内的版本信息与工具版本不匹配，会有三种提示状况：

1. 如果文件内不包含版本信息，会提示以下讯息。

【建议处理方式】读取当前 SDK 版本的默认数据，再打开调试数据文件，这样就可包含差异参数的默认数据，同时也保有文件的调试数据。

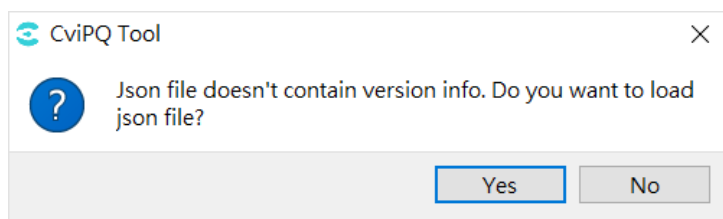


图 5.2: 无版本信息的提示讯息

2. 如果文件内的 SoC 信息与调适工具不匹配，会提示以下讯息。

【建议处理方式】 确认文件和调试工具适用的 SoC 是否一致。

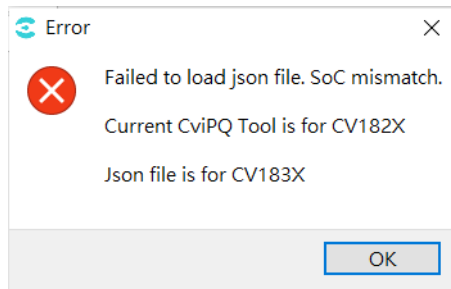


图 5.3: SoC 不匹配的提示讯息

3. 如果文件内的参数版本信息不匹配，会提示以下讯息。

【建议处理方式】 读取当前 SDK 版本的默认数据，再打开调试数据文件，这样就可包含差异参数的默认数据，同时也保有文件的调试数据。

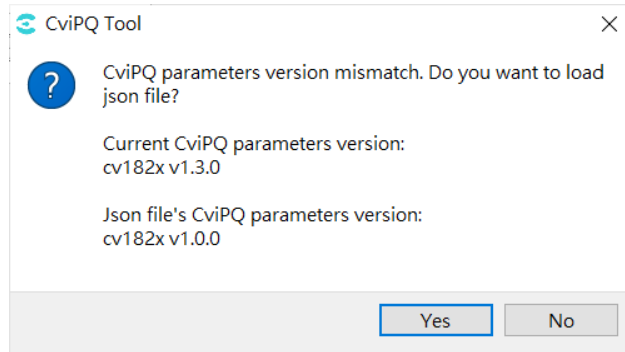


图 5.4: 参数版本不匹配的提示讯息

5.2 为什么工具连接板子会提示 SoC 不匹配？

当工具连接板子时，会确认板子的 SoC 是否和工具匹配？如果不匹配会显示以下错误讯息且中断联机。

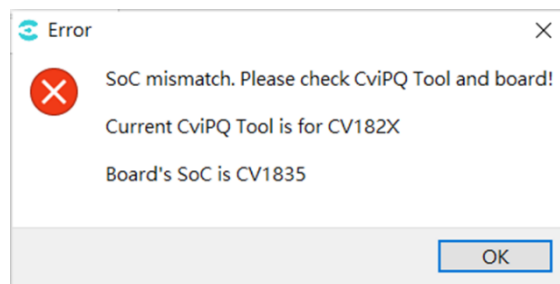


图 5.5: SoC 不匹配的提示讯息

【建议处理方式】 确认工具左上方显示的 SoC 与板子是否一致。

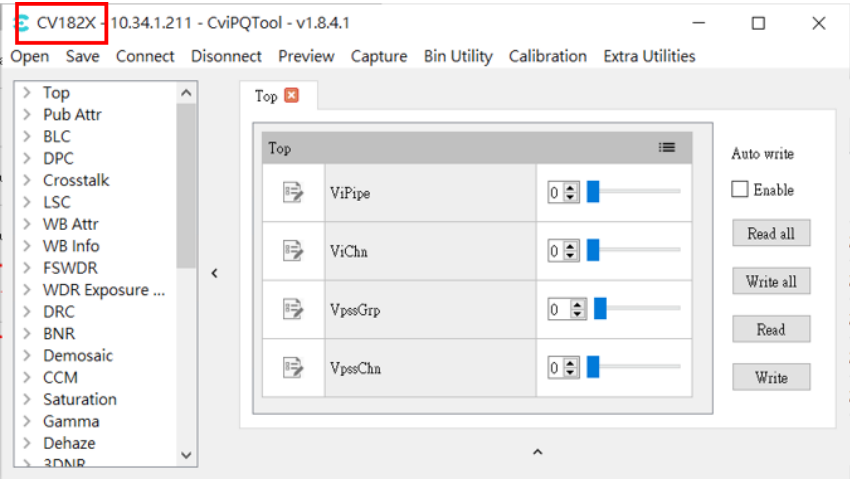


图 5.6: 工具 SoC 信息