

AOV

软件使用指南

文档版本 V1.1

发布日期 2025-12-22

前言

概述

本文档主要介绍 AOV (Always On Video) 方案的基本原理、操作步骤及使用注意事项等，为软件开发人员提供必要的指导。



说明

未经特殊说明，以 XM7206V11A 指代 XM7206 系列所有芯片。FLASH 器件对 AOV 时间影响较大，本文默认以 SPI NOR FLASH 为例。

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2025-09-17	V1.0	首次发布。
2025-12-22	V1.1	新增内核控制 sensor demo，新增实时刷新 osd demo。

目 录

1 AOV 概述	1
1.1 概述	1
2 AOV 功能调试	3
2.1 AOV 硬件测试	3
2.2 sensor 适配	3
2.2.1 sensor api 接口适配	3
2.2.2 sensor 测试	4
2.2.3 sensor 睡眠控制策略	5
2.2.4 Sensor PWDN	5
2.3 AOV sample 介绍	5
2.3.1 API 参考	6
2.3.2 数据类型	12
2.3.3 AOV 内部实现流程	16
2.3.4 Reference design	21
2.3.5 编译运行	21
3 AOV 性能优化	24
3.1 时间优化	24
3.1.1 Flash 性能优化	24
3.1.2 I2C 性能优化	24
3.1.3 量产性能优化	24
3.1.4 调试日志优化	25
3.1.5 Sensor 数据传输优化	25
3.1.6 人车非检测优化	25
3.1.7 媒体通路优化	25

3.1.8 AE 收敛路径优化.....	26
3.2 功耗优化	26
3.2.1 Sensor 功耗优化.....	26
3.2.2 SOC 功耗优化.....	26
4 AOV 功耗测试.....	27
4.1 AOV 测试环境.....	27
4.2 AOV 测试方法.....	28
4.2.1 正常照度场景	28
4.2.2 低照 AINR	28
4.2.3 功耗计算方法	29
4.3 AOV 典型功耗数据	30
4.3.1 2M 线性.....	30
4.3.2 2M AINR	32
4.3.3 4M 线性.....	34
4.3.4 4M AINR	36
5 FAQ	38
5.1 sensor 数据篇.....	38
5.1.1 待机唤醒后出现 isp fe end/ vpss 获取帧失败.....	38
5.1.2 AOV 录制数据出现裂屏等问题.....	38
5.1.3 SENSOR1 由内核驱动控制，SENSOR0 由用户控制	38
5.2 工作周期篇.....	39
5.2.1 AOV 出现工作周期不匹配问题.....	39
5.3 其他篇.....	39
5.3.1 AOV 出现待机无法唤醒	39

插图目录

图 1-1 AOV 工作流程图 1

图 1-2 AOV 工作时间图 2

图 2-1 aov 工作线程处理流程图 17

图 2-2 编码线程处理流程图 18

图 4-1 修改支持 AOV 后的 DEMO 板 27

图 4-2 功耗测试仪 28

图 4-3 单次工作功耗 29

图 4-4 待机功耗 29

图 4-5 多次整机平均功耗 30

图 4-6 2M 线性 isp proc 曝光时间..... 30

图 4-7 2M 线性工作功耗 31

图 4-8 2M 线性待机功耗 31

图 4-9 2M 线性 AOV 周期功耗..... 32

图 4-10 2M AINR proc 曝光时间..... 32

图 4-11 2M AINR 工作功耗 33

图 4-12 2M AINR 待机功耗..... 33

图 4-13 2M AINR AOV 周期功耗 33

图 4-14 4M 线性 isp proc 曝光时间..... 34

图 4-15 4M 线性工作功耗 34

图 4-16 4M 线性待机功耗 35

图 4-17 4M 线性 AOV 周期功耗..... 35

图 4-18 4M AINR proc 曝光时间 36

图 4-19 4M AINR 工作功耗..... 36

图 4-20 4M AINR 待机功耗.....	36
图 4-21 4M AINR AOV 周期功耗	37

仅限深圳市安远威视科技有限公司使用

表格目录

表 4-1 2M 线性 AOV 平均工作功耗 32

表 4-2 2M AINR AOV 平均工作功耗 34

表 4-3 4M 线性 AOV 平均工作功耗 35

表 4-4 4M AINR AOV 平均工作功耗 37

1 AOV 概述

1.1 概述

AOV (Always On Video) 是基于待机唤醒实现的低功耗全天候视频监控技术方案。

在一定时间段里 (比如 1 秒), 摄像机抓拍一次图片 (1fps), 分析图像里是否有特定目标 (比如车, 人), 如果有则摄像机立即切换到正常工作模式 (全帧率录像, 实时视频流等), 如果没有则继续低功耗 1 秒 1 帧模式运行。无特定目标出现时采用短时间工作可以有效降低设备平均功耗, 又能确保不间断拍摄。

图1-1 AOV 工作流程图

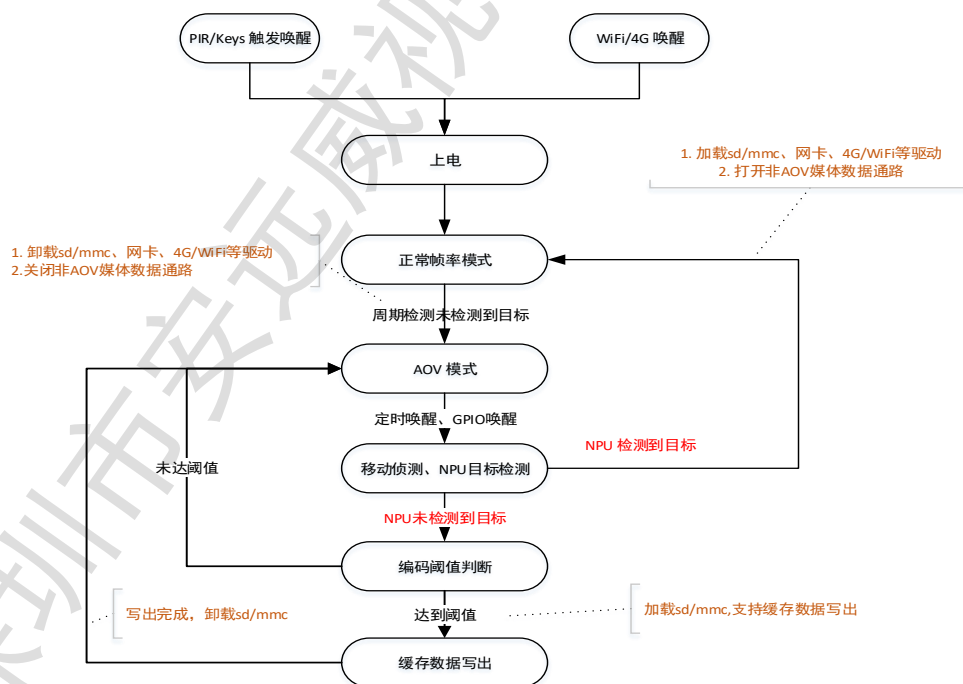
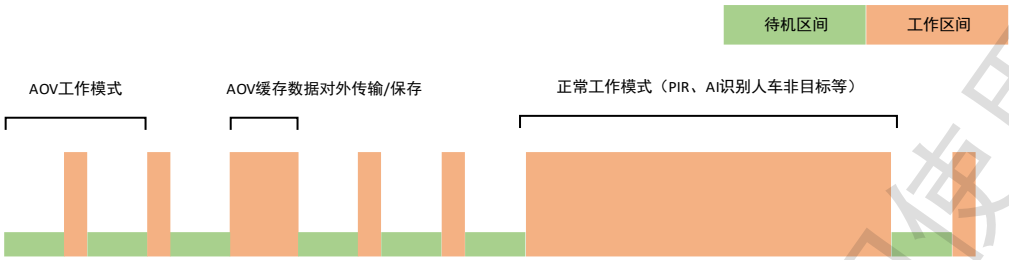


图1-2 AOV 工作时间图



2 AOV 功能调试

2.1 AOV 硬件测试

用户进行 AOV 测试前，需要确认当前单板是否支持 AOV 待机唤醒功能，确认方法：

- 单板上电启动后，进入串口命令行执行 `echo mem >/sys/power/state`。
- 按单板的 PWR_BUTTON 按键。

以上操作一次执行完成后，串口能够正常恢复，即表示该单板支持 AOV 功能，否则不能支持 AOV 功能。（如当前测试未通过，可对比 <AOV 硬件设计指南>，确认相应修改是否完备或咨询硬件相关同事）

2.2 sensor 适配

2.2.1 sensor api 接口适配

以 `sc485sl(sdk/source/gmp/usr/isp/sensor/smartsens_sc485sl/sc485sl.c)` 为例：

用户态实现，睡眠模式使能、退出睡眠模式、低功耗配置（可选）接口。

- `xmedia_s32 sc485sl_resume(xmedia_u32 dev);` // 退出睡眠模式
- `xmedia_s32 sc485sl_standby(xmedia_u32 dev);` // 睡眠模式使能
- `xmedia_s32 sc485sl_low_power(xmedia_u32 dev, xmedia_bool enable);` // 低功耗相关措施（可选）

内核态实现，基于以下全局变量，在 `drv resume` 阶段操作 sensor 寄存器，配置 sensor 数据输出。

```
static const xmedia_sensor_capability g_sc485sl_capability = {
    ...
    .standby_info = {
        .standby_supported = XMEDIA_TRUE,
        .addr_byte_num      = SC485SL_ADDR_BYTE,
        .data_byte_num      = SC485SL_DATA_BYTE,
        .standby_reg_num    = 4,
```

```
.standby_reg_addr = { SC485SL_REG_ADDR_STANDBY, 0x3019, 0x3022,
SC485SL_REG_ADDR_LOW_POWER_STANDBY},
.standby_reg_data = { 0x0, 0xff, 0x03, 0x0f},
.resume_reg_num = 5,
.resume_reg_addr = { XMEDIA_SENSOR_DELAY_FLAG,
SC485SL_REG_ADDR_LOW_POWER_STANDBY, 0x3019, 0x3022,
SC485SL_REG_ADDR_STANDBY},
.resume_reg_data = { 0x5, 0x0, 0x0, 0x01, 0x01},
},};
```

- standby_supported = XMEDIA_TRUE
【AOV 模式使能，sensor 支持 AOV 模式】
- standby_reg_num = 5
【睡眠模式、低功耗模式 sensor 寄存器配置个数】
- standby_reg_addr = {...}
【睡眠模式、低功耗相关配置寄存器，需注意配置顺序，不支持配置延时】
- standby_reg_data = {...}
【睡眠模式、低功耗相关配置寄存器状态】
- resume_reg_num = 6
【退出睡眠模式、低功耗模式 sensor 寄存器配置个数】
- resume_reg_addr = {...}
【退出睡眠模式、低功耗相关配置寄存器，需注意配置顺序，支持配置延时】
- resume_reg_data = {...}
【退出睡眠模式、低功耗相关配置寄存器状态】
- XMEDIA_SENSOR_DELAY_FLAG
【配置寄存器之前延时，单位 ms；仅 resume 支持配置，standby 配置不生效】

2.2.2 sensor 测试

为保证 sensor 以上 3 组接口适配的稳定性，需提前在常电方案进行可靠性测试。

参考 sdk/sample/common/sample_comm_aov.c 接口实现以下接口：

```
sample_aov_sensor_resume
sample_aov_sensor_standby
```

在任意 sample 主进程增加以下函数调用流程：

```
for(xmedia_s32 test_num = 0; test < 100000; test++) {
    sample_aov_sensor_standby(xmedia_u32 isp_pipe);
    usleep(1000*200);
}
```

```
sample_aov_sensor_resume(xmedia_u32 isp_pipe);  
}
```

测试中，无 I2C 报错、无数据恢复失败、无其他媒体通路异常，sensor 初步适配成功。

2.2.3 sensor 睡眠控制策略

2.2.3.1 内核态控制

针对 sensor 消影区过短、应用操作 sensor standby 存在系统调度延时等问题，带来的 sensor standby 不及时导致的图像异常。提供内核态对 sensor 进行睡眠操作的策略。

用户可参考 [sample_aov_set_sensor_mode](#) 实现。

2.2.3.2 用户态控制

针对部分 sensor 可能存在 sensor standby 寄存器配置间隔可能带有延时的问题，内核态控制策略不能满足 sensor 启停策略，提供用户态对 sensor 进行睡眠操作的策略。

用户可参考 `sample_aov_sensor_standby` 实现。（sensor 退出睡眠模式，均在 `misc drv resume` 处操作）

由于该策略下可能存在 sensor standby 不及时的问题，AOV 模式下用户可通过调整帧率和限制最大曝光时间将 sensor 消影区置于安全范围内。

2.2.3.3 Sample 参考

用户可参考 `sample_comm_aov.h` & `sample_comm_aov.c`

注释 `#define APP_STANDBY_SENSOR 1`，即为内核态控制 sensor 进入睡眠模式。
放开即为用户态控制 sensor 进入睡眠模式。

2.2.4 Sensor PWDN

AOV demo 中 sensor 默认使用 I2C 待机模式，如果需要使用 PWDN 待机模式，需将 `source/gmp/drv/misc/include/drv_misc.h` 的 `SUPPORT_SENSOR_PWDN_MODE` 宏定义打开，并在 `drv_misc.c` 适配 sensor 使用的 gpio 管脚和对应寄存器偏移地址。

2.3 AOV sample 介绍

本小节主要介绍 `sample_aov` 的主要流程及参考 demo。

`sdk/sample/aov/sample_aov.c` 单双目参考 demo。

`sdk/sample/comm/sample_comm_aov.c` aov 主要逻辑实现。

2.3.1 API 参考

- `sample_comm_aov_init`: 初始化 AOV, 配置 AOV 参数, 创建线程锁、信号量。
- `sample_comm_aov_exit`: 退出 AOV, 销毁线程锁、信号量。
- `sample_comm_aov_set_work_mode`: 设置工作模式。
- `sample_comm_aov_get_work_mode`: 获取当前工作模式。
- `sample_comm_aov_npu_init`: 初始化 npu。
- `sample_comm_aov_npu_exit`: 退出 npu。
- `sample_comm_aov_npu_create`: 创建 svp task。
- `sample_comm_aov_npu_destroy`: 销毁 svp task。
- `sample_comm_aov_venc_thread_create`: 创建编码取流线程。
- `sample_comm_aov_venc_thread_destroy`: 销毁编码取流线程。
- `sample_comm_aov_work_thread_create`: 创建 AOV 工作线程。
- `sample_comm_aov_work_thread_destroy`: 销毁 AOV 工作线程。
- `sample_comm_aov_misc_init`: misc 初始化, 关联 isp pipe。
- `sample_comm_aov_get_misc_fd`: 获取 misc 句柄。

`sample_comm_aov_init`

【描述】

初始化 AOV, 配置 AOV 参数, 创建线程锁、信号量。

【语法】

```
xmedia_void sample_comm_aov_init(sample_aov_init_param *aov_init_param);
```

【参数】

参数名称	描述	输入/输出
<code>aov_init_param</code>	aov 初始化参数配置。	输入

【返回值】

NA

`sample_comm_aov_exit`

【描述】

退出 AOV，销毁线程锁、信号量。

【语法】

```
xmedia_void sample_comm_aov_exit(xmedia_void);
```

【参数】

NA

【返回值】

NA

sample_comm_aov_set_work_mode

【描述】

设置工作模式。

【语法】

```
xmedia_void sample_comm_aov_set_work_mode(sample_aov_work_mode work_mode);
```

【参数】

参数名称	描述	输入/输出
work_mode	工作模式(正常工作模式、AOV 工作模式)。	输入

【返回值】

NA

sample_comm_aov_get_work_mode

【描述】

获取工作模式。

【语法】

```
xmedia_void sample_comm_aov_get_work_mode(sample_aov_work_mode *work_mode);
```

【参数】

参数名称	描述	输入/输出
*work_mode	工作模式(正常工作模式、AOV 工作模式)。	输出

【返回值】

NA

sample_comm_aov_npu_init

【描述】

初始化 npu、tde。

【语法】

```
xmedia_s32 sample_comm_aov_npu_init(xmedia_void);
```

【参数】

NA

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

sample_comm_aov_npu_exit

【描述】

退出 npu，tde。

【语法】

```
xmedia_void sample_comm_aov_npu_exit(xmedia_s32 handle);
```

【参数】

NA

【返回值】

NA

sample_comm_aov_npu_create

【描述】

创建 svp task。

【语法】

```
xmedia_s32 sample_comm_aov_npu_create(xmedia_s32 *handle, xmedia_svp_task_cfg *task_cfg)
```

【参数】

参数名称	描述	输入/输出
*handle	svp 句柄。	输出
*task_cfg	svp 模型相关参数。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

sample_comm_aov_npu_destroy

【描述】

销毁 svp task。

【语法】

```
xmedia_s32 sample_comm_aov_npu_destroy(xmedia_s32 handle);
```

【参数】

参数名称	描述	输入/输出
handle	svp 句柄。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

sample_comm_aov_venc_thread_create

【描述】

创建编码取流检测线程。

【语法】


```
xmedia_s32 sample_comm_aov_venc_thread_create(xmedia_void);
```

【参数】

NA, 由 aov_init_param 统一传递。

【返回值】

返回值	描述
0	成功。
非 0	失败, 其值为错误码。

sample_comm_aov_venc_thread_destroy

【描述】

销毁编码取流线程。

【语法】

```
xmedia_s32 sample_comm_aov_venc_thread_destroy(xmedia_void);
```

【参数】

NA, 由 aov_init_param 统一传递。

【返回值】

返回值	描述
0	成功。
非 0	不涉及。

sample_comm_aov_work_thread_create

【描述】

创建 AOV 工作线程。

【语法】

```
xmedia_s32 sample_comm_aov_work_thread_create(xmedia_void);
```

【参数】

NA

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

sample_comm_aov_work_thread_destroy**【描述】**

销毁 AOV 工作检测线程。

【语法】

```
xmedia_s32 sample_comm_aov_work_thread_destroy(xmedia_void);
```

【参数】

NA，由 aov_init_param 统一传递。

【返回值】

返回值	描述
0	成功。
非 0	不涉及。

sample_comm_aov_misc_init**【描述】**

misc 驱动关联 isp pipe，控制 sensor 进行 resume & standby。

【语法】

```
xmedia_s32 sample_comm_aov_misc_init(misc_aov_isp_info *isp_info);
```

【参数】

参数名称	描述	输入/输出
isp_info	isp pipe num & pipe。	输入

【返回值】

返回值	描述
0	成功。
非 0	参数错误。

sample_comm_aov_get_misc_fd

【描述】

获取 misc 句柄。

【语法】

```
xmedia_s32 sample_comm_aov_get_misc_fd (xmedia_void);
```

【参数】

NA。

【返回值】

misc 句柄，用于 osd 刷新接口。

2.3.2 数据类型

- [sample_aov_normal_npu_notice](#): 正常模式，返回 npu 检测结果回调。
- [sample_aov_normal_resume](#): AOV 模式切正常模式时，用户业务恢复逻辑。
- [sample_aov_normal_suspend](#): 正常模式切 AOV 模式时，用户业务暂停逻辑。
- [sample_aov_isp_param](#): isp pipe & pipe num, standby & resume sensor。
- [sample_aov_vpss_normal_only_param](#): 只在正常模式使用的 vpss ochn。
- [sample_aov_vpss_npu_param](#): npu 检测句柄及 npu 检测数据来源。
- [sample_aov_venc_param](#): 编码通道信息。
- [sample_aov_work_mode](#): 工作模式（正常模式，AOV 模式）。

sample_aov_normal_npu_notice

【说明】

正常模式，返回 npu 检测结果回调。

【定义】

```
typedef xmedia_void (*sample_aov_normal_npu_notice)(sample_aov_work_mode);
```

【成员】

NA

sample_aov_normal_resume

【说明】

AOV 模式切正常模式时，用户业务恢复逻辑。用于客户自己实现。

【定义】

```
typedef xmedia_void (*sample_aov_normal_resume)(xmedia_void);
```

【成员】

NA

sample_aov_normal_suspend

【说明】

正常模式切 AOV 模式时，用户业务暂停逻辑。用于客户自己实现。

【定义】

```
typedef xmedia_void (*sample_aov_normal_suspend)(xmedia_void);
```

【成员】

NA

sample_aov_isp_param

【说明】

isp pipe & pipe num, standby & resume sensor。

【定义】

```
typedef struct sample_aov_isp_param {  
    xmedia_s32 isp_pipe_num;  
    xmedia_s32 isp_pipe[VI_MAX_DEV_NUM];  
} sample_aov_isp_param;
```

【成员】

成员名称	描述
isp_pipe_num	isp 通道个数。
isp_pipe	isp 通道号。

sample_aov_vpss_normal_only_param

【说明】

只在正常模式使用的 vpss ochn，正常模式切到 AOV 模式 disable，AOV 切回正常模式 enable。

【定义】

```
typedef struct sample_aov_vpss_normal_only_param {  
    xmedia_s32 vpss_pipe_num;  
    xmedia_s32 vpss_pipe[VPSS_MAX_PIPE_NUM];  
    xmedia_s32 vpss_ochn_num[VPSS_MAX_PIPE_NUM];  
    xmedia_s32 vpss_ochn[VPSS_MAX_PIPE_NUM][VPSS_MAX_OCHN_NUM];  
} sample_aov_vpss_normal_only_param;
```

【成员】

成员名称	描述
vpss_pipe_num	vpss pipe 个数。
vpss_pipe	vpss pipe id。
vpss_ochn_num	vpss pipe 对应 vpss ochn 个数。
vpss_ochn	vpss pipe 对应 vpss ochn id。

sample_aov_vpss_npu_param

【说明】

npu 检测句柄及 npu 检测数据来源。

【定义】

```
typedef struct sample_aov_vpss_npu_param {  
    xmedia_s32 npu_detect_num;  
    xmedia_s32 npu_svp_handle[VPSS_MAX_PIPE_NUM];  
    xmedia_bool en_switch_mode[VPSS_MAX_PIPE_NUM];  
    xmedia_s32 vpss_pipe[VPSS_MAX_PIPE_NUM];  
    xmedia_s32 vpss_ochn[VPSS_MAX_PIPE_NUM];  
} sample_aov_vpss_npu_param;
```

【成员】

成员名称	描述
npu_detect_num	npu svp task 个数。

成员名称	描述
npu_svp_handle	svp handle。
en_switch_mode	正常模式与 AOV 模式切换时，svp 检测结果是否切换成仅目标输出。
vpss pipe	svp 数据来源 vpss pipe id。
vpss ochn	svp 数据来源 vpss ochn id。

sample_aov_venc_param

【说明】

编码通道信息。

【定义】

```
typedef struct sample_aov_venc_param {
    xmedia_s32  venc_chn_num;
    xmedia_bool en_aov[VENC_MAX_CHN_NUM];
    xmedia_s32  venc_chn[VENC_MAX_CHN_NUM];
    xmedia_s32  chn_max_size[VENC_MAX_CHN_NUM];
    xmedia_s32  chn_threshold_size[VENC_MAX_CHN_NUM];
} sample_aov_venc_param;
```

【成员】

成员名称	描述
venc_chn_num	venc 通道数。
en_aov	AOV 模式下是否继续编码。
venc_chn	venc 通道号。
chn_max_size	venc 通道 stream buf 大小。
chn_threshold_size	venc 通道缓存阈值。 (chn_max_size - cur_used < chn_threshold_size)。

sample_aov_work_mode

【说明】

工作模式（正常模式，AOV 模式）。

【定义】

```
typedef enum sample_aov_work_mode {  
    MEDIA_WORK_AOV = 0,  
    MEDIA_WORK_NORMAL,  
} sample_aov_work_mode;
```

【成员】

成员名称	描述
MEDIA_WORK_AOV	AOV 模式。
MEDIA_WORK_NORMAL	正常工作模式。

2.3.3 AOV 内部实现流程

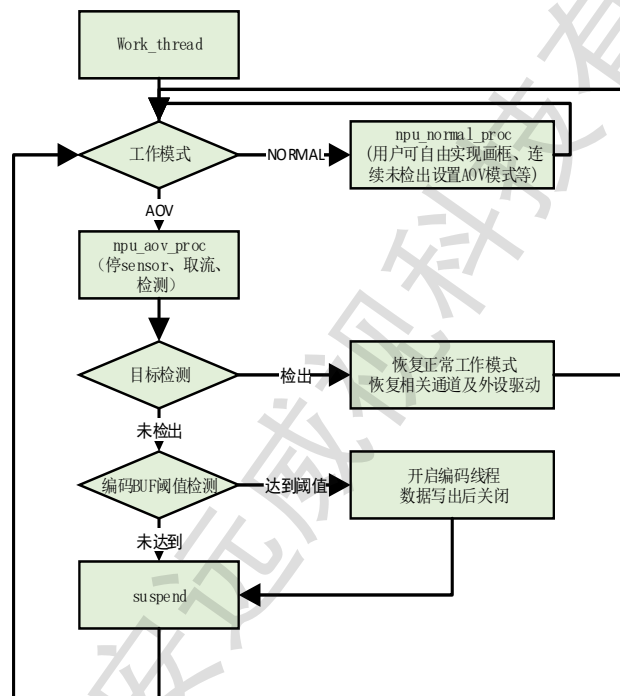
本小节主要介绍 AOV 功能实现内部结构，用户可根据实际需求，修改或新增。

- [sample_aov_work_thread](#): AOV 工作线程。(正常工作模式用于 AI 检测)
- [sample_aov_venc_stream_thread](#): 编码线程。(AOV 模式长期处于 IDEL 状态)
- [sample_aov_npu_normal_proc](#): 正常工作模式，用于 NPU 检测处理流程。(用户可根据实际需求，修改新增)
- [sample_aov_npu_aov_proc](#): AOV 工作模式，NPU 检测处理流程。(调用流程不建议调整)
- [sample_aov_work_normal_pipe_resume](#): AOV 工作模式切正常工作模式下，媒体通路恢复正常工作模式相关逻辑。
- [sample_aov_work_normal_pipe_suspend](#): 正常工作模式切 AOV 模式下，媒体通路拆解关闭，仅保留 AOV 相关媒体通路。
- [sample_aov_work_aov_save_resume](#): AOV 模式，编码数据写出前需加载相应设备驱动，并开启编码写出线程。
- [sample_aov_work_aov_save_suspend](#): AOV 模式，等待编码数据写出完成后，卸载相应驱动。
- [sample_aov_venc_set_stream_save_start](#): 开启编码线程缓存数据写出。
- [sample_aov_venc_get_stream_save_stop](#): 获取编码线程写出完成。(阻塞接口)
- [sample_aov_venc_save_stream](#): 编码缓存数据写出。
- [sample_aov_venc_get_aov_buf_full](#): 判断编码缓存是否达到阈值。

- `sample_aov_venc_clear_buf`: 清除 AOV 编码通道缓存数据。
- `sample_aov_set_suspend_time`: 待机周期校准。
- `sample_aov_suspend`: 配置待机参数，进入待机。
- `sample_aov_sensor_resume`: sensor 数据恢复（退出 sensor 低功耗等）。
- `sample_aov_sensor_standby`: sensor 数据暂停（配置 sensor 低功耗等）。
- `sample_aov_npu_set_work_mode`: 设置 NPU 检测结果输出模式。
- `sample_aov_set_sensor_mode`: 设置 sensor 输出模式。（内核 standby sensor）

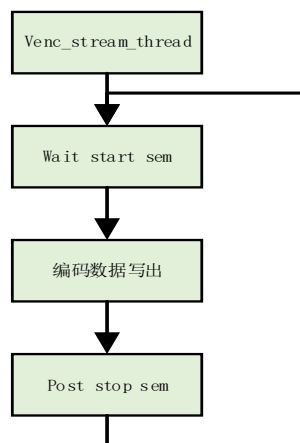
`sample_aov_work_thread`

图2-1 aov 工作线程处理流程图



sample_aov_venc_stream_thread

图2-2 编码线程处理流程图



sample_aov_npu_normal_proc

【说明】正常工作模式，用于 NPU 检测处理流程。

在正常模式中，用户可参考该接口流程实现画框、配置正常模式 N 次未检测到目标进入 AOV 等功能。

sample_aov_npu_aov_proc

【说明】AOV 工作模式，NPU 检测处理流程。

1. Wait FE_END，等待帧完成。
2. 停止 sensor 输出并将 sensor 置于低功耗模式。
3. NPU 检测，输出检测结果。

sample_aov_work_normal_pipe_resume

【说明】AOV 工作模式切正常工作模式下，媒体通路恢复正常工作模式相关逻辑。

1. 切换 NPU 检测模式，退出仅输出检测结果模式。

可参考：[sample_aov_npu_set_work_mode](#)

2. 加载或绑定正常工作模式需要的外部设备驱动或 gmp 驱动。
3. 恢复正常工作模式下需要的媒体通道。
4. 恢复 sensor 数据输出。

sample_aov_work_normal_pipe_suspend

【说明】正常工作模式切 AOV 模式下，媒体通路拆解关闭，仅保留 AOV 相关媒体通路。

1. 停止 sensor 数据输出。
2. 编码数据完全写出完成，关闭 AOV 不涉及的媒体通路。
3. 切换 NPU 检测模式，进入仅输出检测结果模式。
4. 卸载或解绑定正常工作模式需要的外部设备驱动或 gmp 驱动。

sample_aov_work_aov_save_resume

【说明】AOV 模式，编码数据写出前需加载相应设备驱动，并开启编码写出线程。

1. 加载或绑定外部设备驱动，支持编码缓存数据写出。（如 sdcard 驱动）
2. 打开编码缓存数据写出线程。

sample_aov_work_aov_save_suspend

【说明】AOV 模式，等待编码数据写出完成后，卸载相应驱动。

1. 等待编码缓存数据写出结束。（阻塞接口，需等编码数据写完再卸载驱动）
2. 卸载或解绑外部设备驱动。（如 sdcard 驱动）

sample_aov_venc_set_stream_save_start

【说明】开启编码线程缓存数据写出。

设置 g_venc_steram_save_start_sem 信号量启动编码线程。

sample_aov_venc_get_stream_save_stop

【说明】获取编码线程写出完成。（阻塞接口）

等待编码写出完成信号量 g_venc_steram_save_stop_sem 。

sample_aov_venc_save_stream

【说明】编码缓存数据写出。（用户根据需求自行修改）

sample_aov_venc_get_aov_buf_full

【说明】判断编码缓存是否达到阈值。

1. 判断各编码通道 $chn_max_size - cur_chn_used \leq chn_threshold_size$ 。
2. 判断各编码通道帧数是否超过配置的编码 buf 最大帧数。

3. 阈值自适应，各通道阈值刷新成历史最大帧大小。

sample_aov_venc_clear_buf

【说明】清除 AOV 编码通道缓存数据。(避免缓存数据影响 AOV 周期)

sample_aov_set_suspend_time

【说明】待机周期校准。详细描述参考 [sample_aov_suspend](#)。

优化 POR – rtc drv resume 间隔误差,计算 app 耗时，避免 rtc 配置超时。

sample_aov_suspend

【说明】配置待机参数，进入待机。

1. RTC 设置待机周期分别位于 rtc drv resume、rtc drv suspend 阶段。
2. xmedia_pm_set_attr 接口调用一次，即在两处均配置周期，否则仅在 rtc drv resume 阶段配置。
3. 当 app 超时，需调用 xmedia_pm_set_attr。

sample_aov_sensor_resume

【说明】sensor 数据恢复（退出 sensor 低功耗等）。

1. sensor 退出 mipi 高阻态。(根据 sensor 特性可选)
2. sensor 退出低功耗模式。(根据 sensor 特性可选)
3. sensor 数据恢复。

sample_aov_sensor_standby

【说明】sensor 数据暂停（配置 sensor 低功耗等）。

1. sensor 数据暂停。
2. sensor 进入低功耗模式。(根据 sensor 特性可选)
3. sensor 设置 mipi 高阻态。(防止 sensor 电流倒灌。根据 sensor 特性可选)

sample_aov_npu_set_work_mode

【说明】设置 NPU 检测结果输出模式。

1. 可配置仅目标检测结果输出。(AOV 模式适用，优化检测耗时)
2. 可配置目标检测结果 + 目标框结果。(正常模式适用)

sample_aov_set_sensor_mode

【说明】设置 sensor 输出模式。

1. aov 模式，输出一帧后 sensor 进入睡眠模式。
2. normal 模式，连续输出。

2.3.4 Reference design

2.3.4.1 OSD 实时刷新

由于 AOV 场景中，系统唤醒时，就开始通知 sensor 开始曝光，如果 sensor 开始传输图像时，OSD 还未绘制好，就会出现打在图像上的日期时间仍是上一个周期绘制的 OSD，导致视频时间与实际时间有出入。为确保日期时间 OSD 及时更新，现将日期时间 OSD 功能实现放到 misc 驱动中，在 sensor 曝光前绘制好 OSD。

- misc 驱动位置: source/gmp/drv/misc, 相关文件: drv_misc.h、drv_osd_draw_time.h、drv_osd_draw_time.c。
- 时间日期 OSD 功能默认关闭，如需打开请放开 source/gmp/drv/misc/include/drv_misc.h 的 SUPPORT_OSD_TIME 宏定义。为节省整体工作时间和内存，最好将一个日期时间 OSD 绑定所有的视频编码。
- source/gmp/drv/misc/src/drv_osd_draw_time.c 文件实现 osd 图像绘制，使用 8x8 点阵字体，支持 0-9 数字、冒号、空格和横杠等字符绘制，如需支持其他字符，用户可自行扩展。
- 用户态应用调用 misc 驱动使能日期时间 OSD 功能相关代码见 AOV demo sample_osd_time.c 相关代码。另外，退出 aov 模式，回到 normal 模式后，需在应用中定时通知 misc 刷新日期时间 OSD(见 sample_aov.c 文件调用 sample_osd_time_update 函数的代码)。

注意：

- misc 驱动只实现日期时间 OSD，对于其他如 logo 叠加，需用户自行在用户态实现；
- 用于日期时间 OSD 的 region handle 请勿在用户态做其他绘图，否则会出现绘图异常问题，因为用户态和驱动都在用同一 handle 进行绘图。

2.3.5 编译运行

2.3.5.1 选择内核编译配置参数

1. 选择快启编译配置编译。

Eg:选择 cp configs/ xm7206v11a / xm7206v11arba_linux-5.10_quickstart_cfg.mk ./cfg.mk

2. 选择非快启编译配置编译。

用户可选择非快启编译配置编译，需自行修改内核配置，确保 usb、eth 等驱动在进入 AOV 模式前卸载或解绑，避免影响 AOV 工作耗时及正常待机唤醒。

2.3.5.2 gmp 驱动配置

用户可通过修改以下文件，筛选掉当前 aov 应用不涉及的媒体驱动。若仅在正常工作模式需要，可在进入 AOV 模式前卸载或解绑对应媒体驱动。

sdk/source/gmp/drv/init/xmedia_drv_init.c

Eg:

```
#if 0
    vo_driver_init();
    gfbg_driver_init();
    acomm_driver_init();
    aiao_driver_init();
    ao_driver_init();
    ai_driver_init();
    aenc_driver_init();
    adec_driver_init();
#endif
```

2.3.5.3 配置 rootfs 启动项

- 编译完成后修改启动脚本(用户亦可修改内核配置，不编译不使用的驱动)。

sdk/out/xm7206v11a/rootfs/etc/init.d/ S00devs中,注释以下流程:

```
#modules_list=$(echo $(find /lib/modules/$(uname -r) -name *.ko))
#for mod in ${modules_list}; do
    #echo "Load $mod"
    #modprobe $(basename $mod)
#done
```

- 增加 sdcard 或其他必要驱动加载。

```
insmod /lib/modules/5.10.130/kernel/drivers/mmc/core/mmc_core.ko
insmod /lib/modules/5.10.130/kernel/drivers/mmc/core/mmc_block.ko
insmod /lib/modules/5.10.130/kernel/drivers/mmc/host/sdhci.ko
insmod /lib/modules/5.10.130/kernel/drivers/mmc/host/sdhci-pltfm.ko
insmod /lib/modules/5.10.130/kernel/lotus/drivers/mmc/host/sdhci-lotus.ko
insmod /lib/modules/5.10.130/kernel/drivers/mmc/core/pwrseq_emmc.ko
insmod /lib/modules/5.10.130/kernel/drivers/mmc/core/pwrseq_simple.ko
```

- sdk/sample/sample_base.mk 中选择 sensor 类型。

- 拷贝 sample_aov 及模型文件到 rootfs 分区。

将 sample_aov 拷贝到 sdk/out/ xm7206v11a /rootfs/usr/bin, 将模型文件、pq.bin 等文件拷贝到 sdk/out/ xm7206v11a/rootfs/usr/param, 重新打包 rootfs。

- make jffs2_clean; make jffs2

3 AOV 性能优化

3.1 时间优化

本小节主要描述如何对 AOV 工作耗时进行优化，请用户根据实际应用场景选择适当的优化策略。

3.1.1 Flash 性能优化

XM7206V11A 在 AOV 唤醒过程中，从 flash 中读取 uboot、auxcode、ddr param，提高 flash 的频率能够有效缩短 AOV 工作耗时。请客户根据 flash 型号选择 flash 频率，修改方式如下：

可参考<OTP API 参考>实现。（仅支持操作一次，操作后无法还原，请谨慎操作。）

- xmedia_otp_init
- xmedia_otp_program_enable
- xmedia_otp_fmc_freq_sel
- xmedia_otp_exit

3.1.2 I2C 性能优化

XM7206V11A 内核 I2C 频率默认配置 100kHz。用户可修改 dts 将 I2C 频率提高到 400kHz，以缩短 AOV 唤醒过程中操作 sensor 寄存器耗时。（优化耗时与 sensor 数据恢复寄存器个数相关）

修改方式如下：

sdk/source/kernel/linux-5.10.y/lotus/machine/xmorca/dts/xmorca.dts

3.1.3 量产性能优化

用户可关闭串口烧录功能，优化耗时 50ms。修改方式如下：

可参考<OTP API 参考>实现。（仅支持操作一次，操作后无法还原，请谨慎操作。）

- xmedia_otp_init
- xmedia_otp_program_enable
- xmedia_otp_uart_burn_mode_sel

- xmedia_otp_exit

3.1.4 调试日志优化

3.1.4.1 Bootrom 日志关闭

可参考<OTP API 参考>实现。(仅支持操作一次, 操作后无法还原, 请谨慎操作。)

- xmedia_otp_init
- xmedia_otp_program_enable
- xmedia_otp_bootrom_debug_disable
- xmedia_otp_exit

3.1.4.2 Auxcode 日志关闭

用户可通过以下操作, 关闭 Auxcode 阶段的调试日志:

```
sdk/source/bootloader/u-boot-2020.01/configs/xmorca_quickstart_defconfig  
CONFIG_AUXCODE_LOG_CTRL=0
```

3.1.4.3 Kernel 日志关闭

用户可通过修改 bootargs loglevel=0 或 串口输入 dmesg -n 1 , 来关闭 Kernel 日志。

3.1.5 Sensor 数据传输优化

Sensor 配置为高速率低帧率 (60fps 序列跑 30fps), 对比低速率相同帧率 (30fps 序列跑 30fps), 能够有效优化单帧传输时间。

3.1.6 人车非检测优化

当前 AOV demo 引入以下两种方法, 优化检测耗时, 用户可按实际需求进行选择。

- 目标检测前先进行 ipe 移动侦测。若移动侦测未识别到移动目标, 则不进行目标检测; 若移动侦测识别到移动目标, 进行目标检测再次过滤。
- AOV 工作场景下, 目标检测可以只输出检测结果, 不输出检测目标坐标, 降低后处理时间。正常工作模式再切换可输出坐标点。

可参考: [sample_aov_npu_set_work_mode](#)

3.1.7 媒体通路优化

用户创建媒体通路时, 需尽量使用媒体通路在线或低延时, 降低媒体通路耗时。

3.1.8 AE 收敛路径优化

用户在调节 PQ 参数时，可在照度良好的场景下，不影响图像效果的前提下，适当加大 sensor 模拟增益倍数，减少曝光时间，从而降低整体通路耗时。接口请参考：

```
xmedia_s32 xmedia_ae_set_route_attr(xmedia_u32 pipe, const xmedia_isp_ae_route
*ae_route_attr);
```

3.2 功耗优化

本小节主要描述如何对工作功耗、待机功耗进行优化，请用户根据实际应用场景选择适当的优化策略。

3.2.1 Sensor 功耗优化

Sensor 一般可以通过 I2C 操作寄存器、拉低 sensor PWDN 等方式让 sensor 进入待机状态，用户可对比两者待机功耗差异，选用功耗更优的方案或两种方案结合的方式，进行待机操作。

3.2.2 SOC 功耗优化

用户需在待机前卸载 AOV 模式下不使用的设备驱动（sd 卡、usb、eth 等），减少 AOV 工作周期耗时。可参考：

```
static xmedia_s32 sample_aov_drv_sdcard_unload();
```

4 AOV 功耗测试

4.1 AOV 测试环境

图4-1 修改支持 AOV 后的 DEMO 板

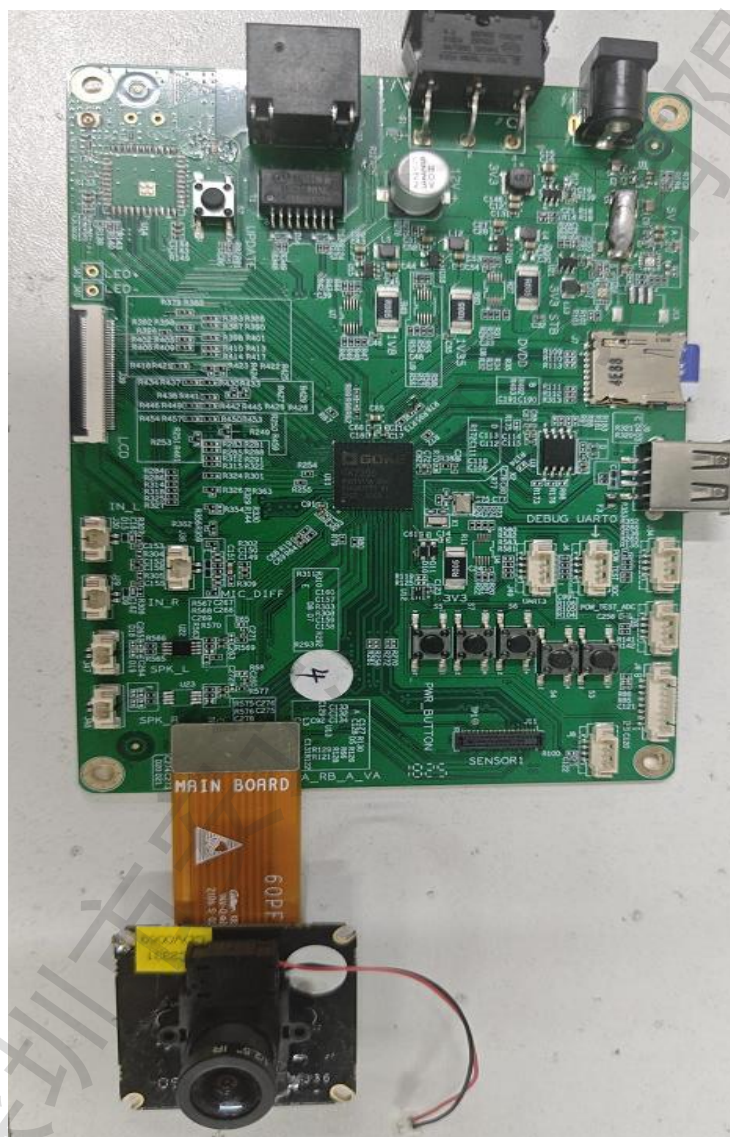


图4-2 功耗测试仪



测试前需确保以下注意点：

- 确保 sdcard 已插入，并能正常读写
- 确保 sensor 已调焦

4.2 AOV 测试方法

4.2.1 正常照度场景

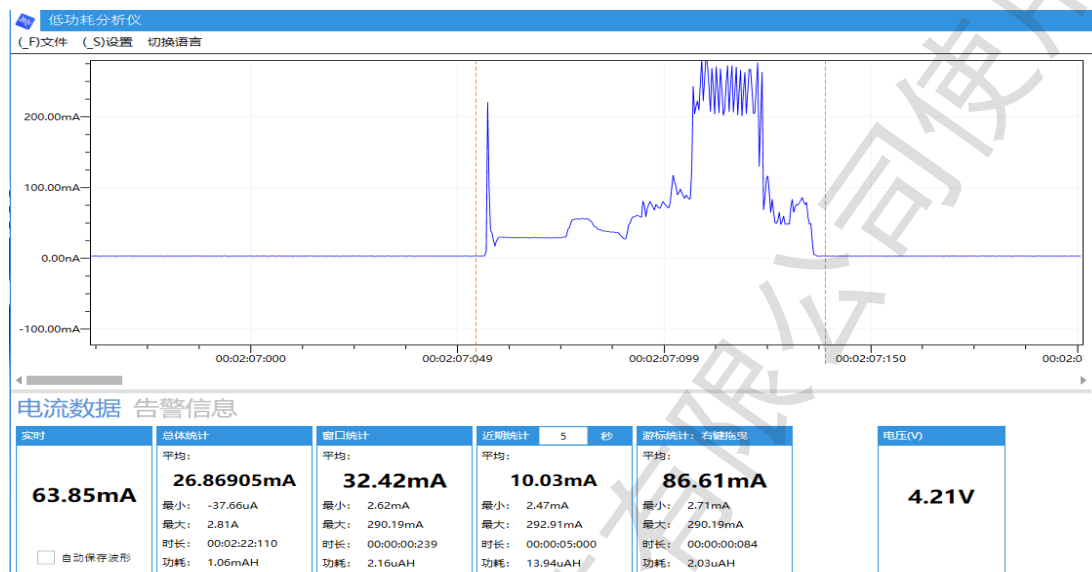
用户可将镜头朝向光照稳定、光照良好的环境下，将正常工作模式适当延长，待收敛结束之后 `cat /proc/umap/isp` 读出当前 sensor 曝光时间。待进入 AOV 后可通过框选功耗波形图的方式得到相应的功耗数据。

4.2.2 低照 AINR

用户可将镜头捂住或置于暗室环境下，将正常工作模式适当延长，待收敛结束之后 `cat /proc/umap/isp` 读出当前 sensor 曝光时间，确认为当前帧率下最大曝光时间。待进入 AOV 后可通过框选功耗波形图的方式得到相应的功耗数据。

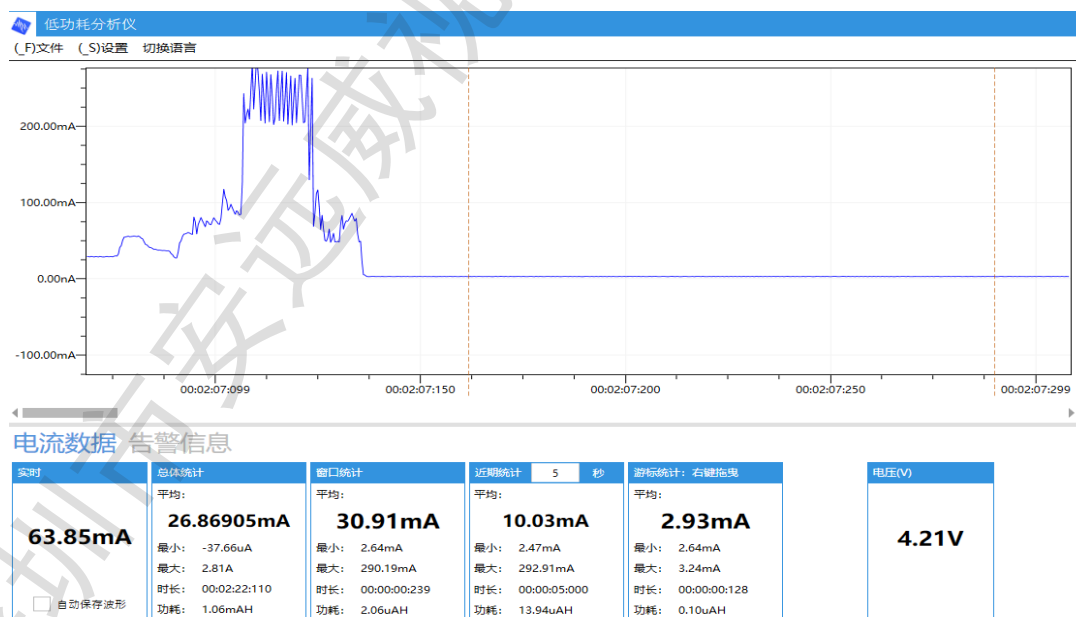
4.2.3 功耗计算方法

图4-3 单次工作功耗



- AOV 工作耗时: 84ms
- AOV 工作平均功耗: $4.21 \times 86.61 = 364.63\text{mW}$

图4-4 待机功耗



- 待机功耗: $2.93 \times 4.21 = 12.34\text{mW}$

图4-5 多次整机平均功耗



- 平均功耗 (不含写 sdcard): $10.04 \times 4.21 = 42.27 \text{ mW}$

4.3 AOV 典型功耗数据

以下功耗数据, 2M 使用 sc235hai 60fps 序列跑 25fps、12fps。4M 线性采用 sc485sl (2560*1440) 60fps 序列跑 25fps, 4M AINR 采用 sc485sl (2688*1520) 30fps 跑 12fps。

4.3.1 2M 线性

图4-6 2M 线性 isp proc 曝光时间

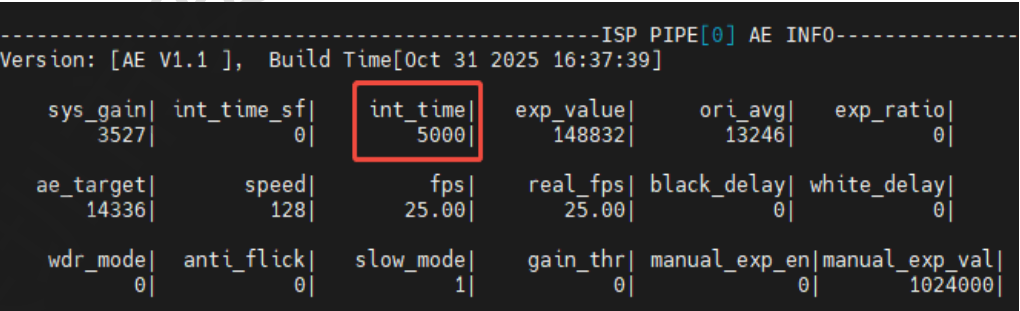


图4-7 2M 线性工作功耗

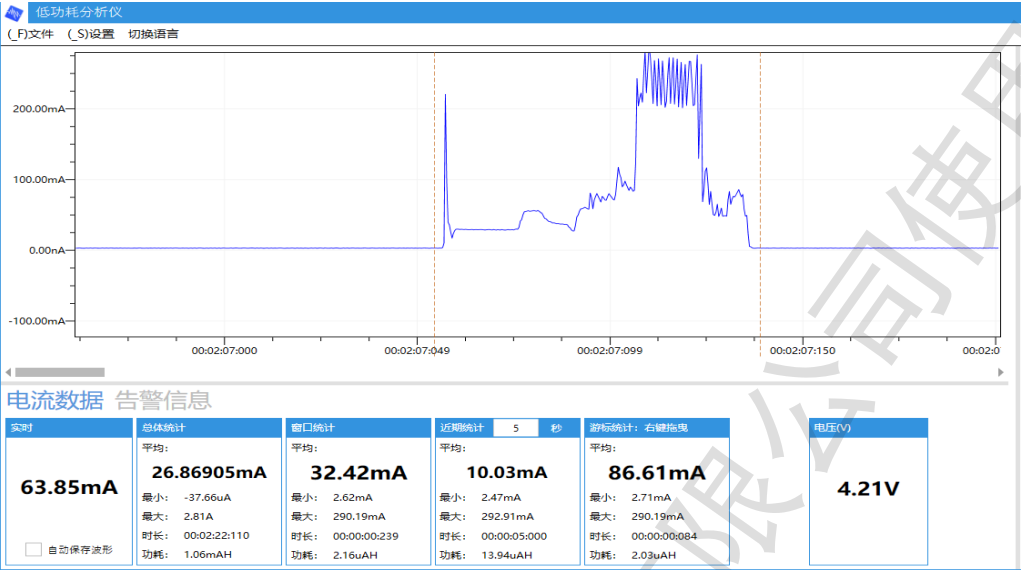


图4-8 2M 线性待机功耗

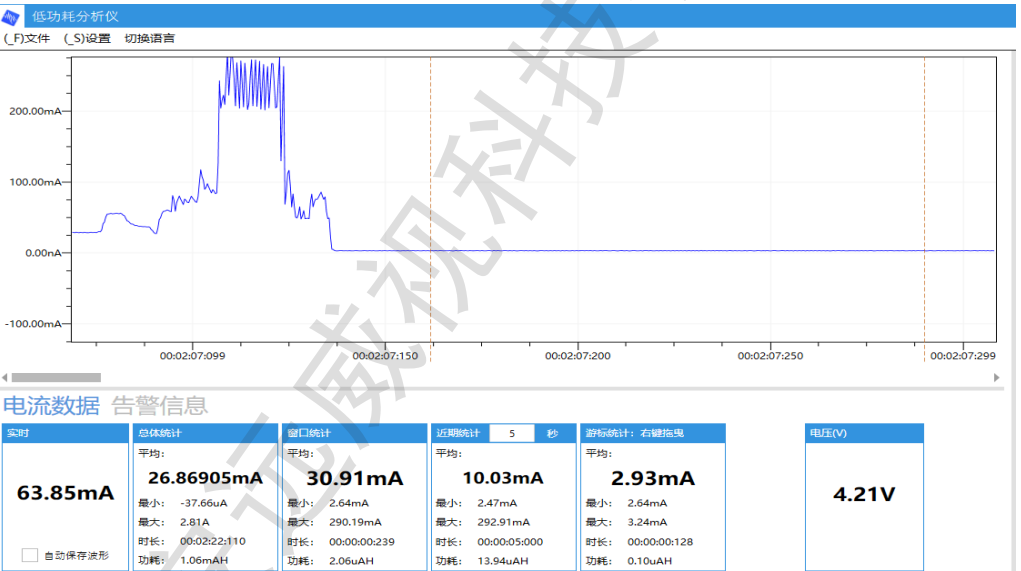


图4-9 2M 线性 AOV 周期功耗

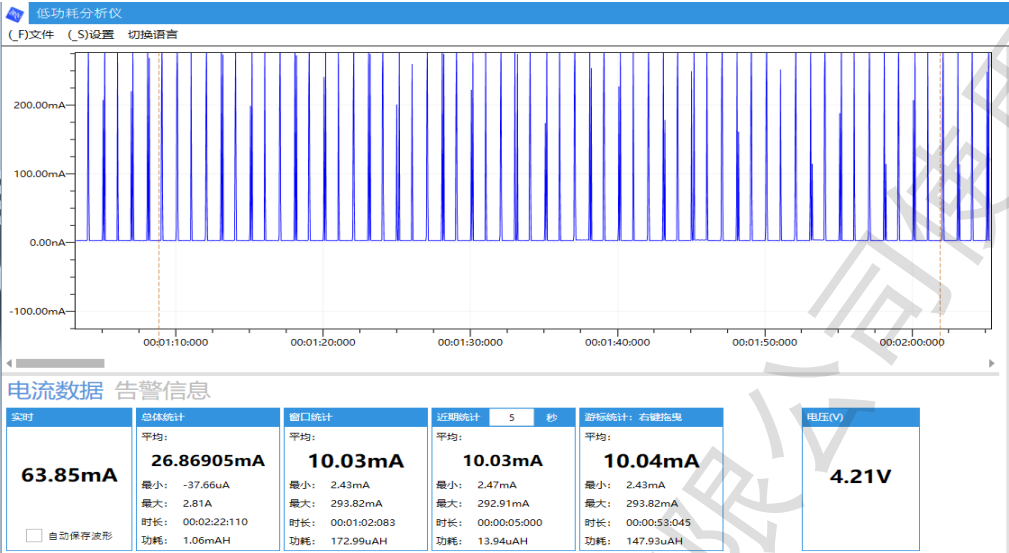


表4-1 2M 线性 AOV 平均工作功耗

工作平均功耗 (mW)	工作周期 (ms)	待机平均功耗 (mW)	待机周期 (ms)	整机平均功耗 (mW)
364.6	84	12.34	916	41.92984
			2916	22.20328
			14916	14.31266

4.3.2 2M AINR

图4-10 2M AINR proc 曝光时间

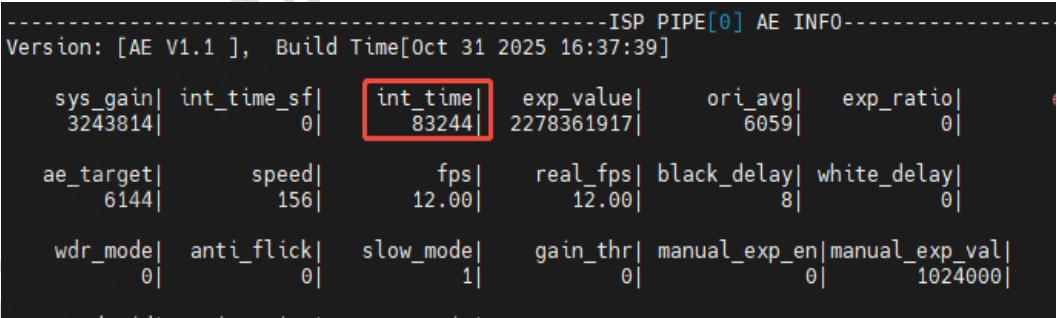


图4-11 2MAINR 工作功耗

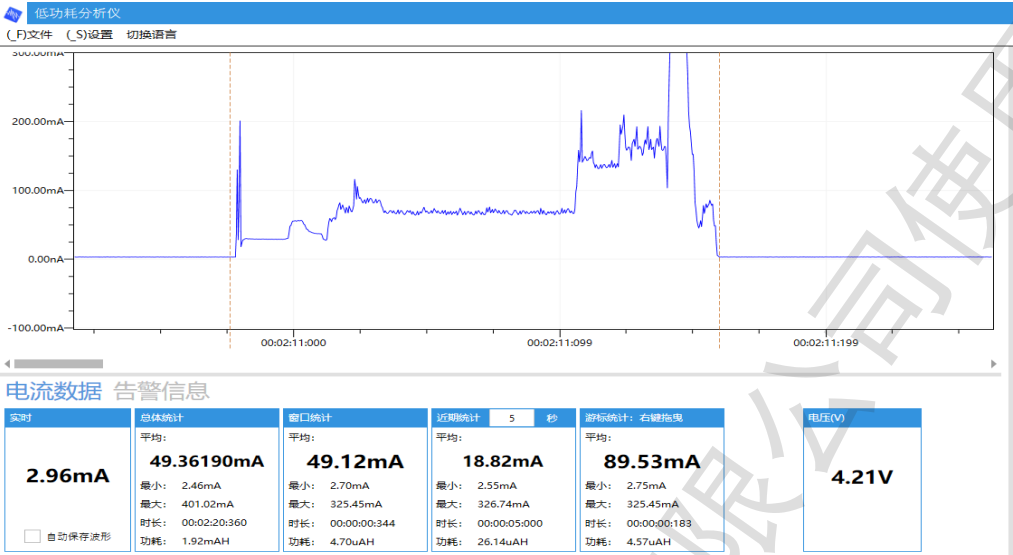


图4-12 2MAINR 待机功耗

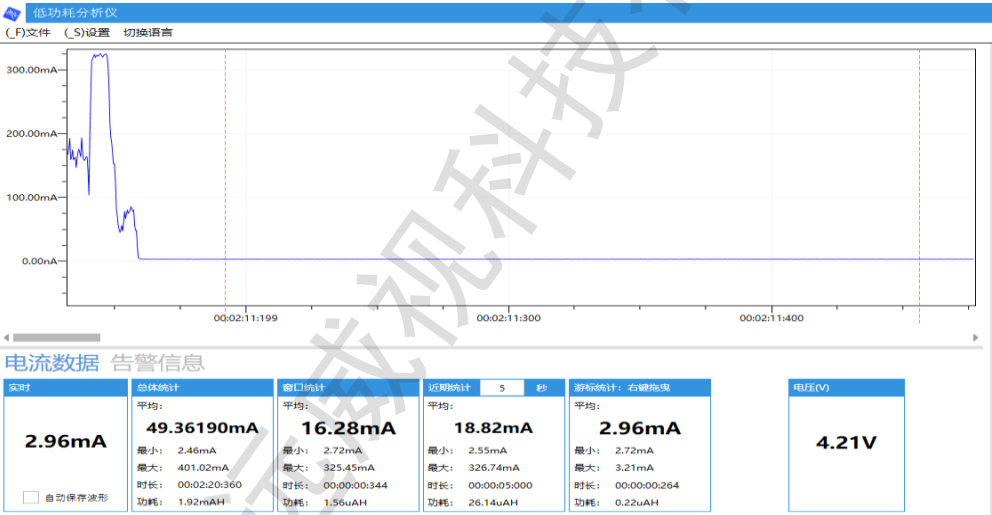


图4-13 2MAINRAOV 周期功耗

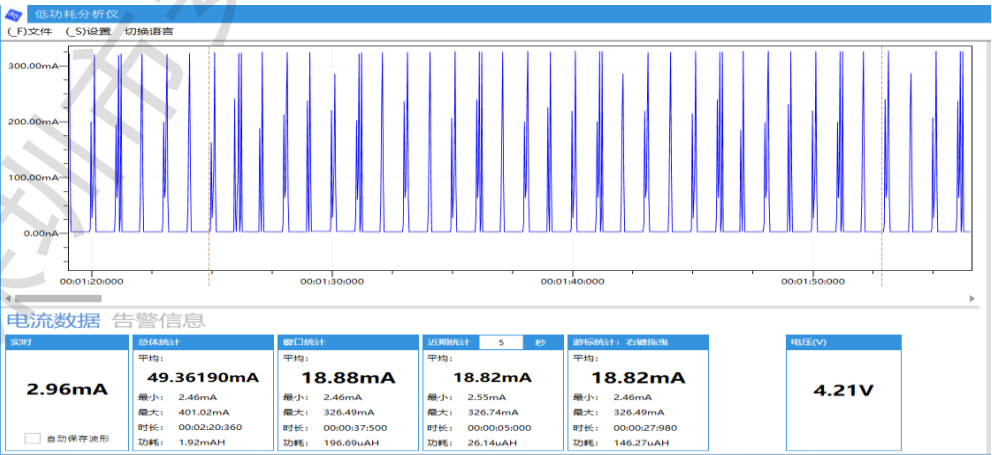


表4-2 2MAINRAOV 平均工作功耗

工作平均功耗 (mW)	工作周期 (ms)	待机平均功耗 (mW)	待机周期 (ms)	整机平均功耗 (mW)
376.92	183	12.46	814	79.15618
			2814	34.69206
			14814	16.90641

4.3.3 4M 线性

图4-14 4M 线性 isp proc 曝光时间

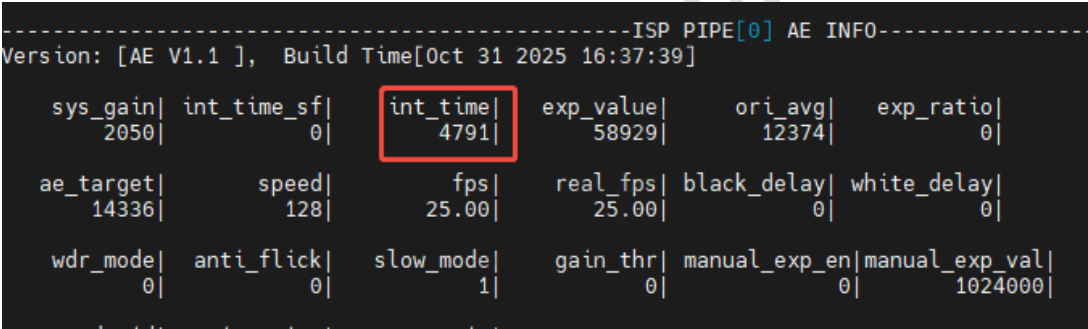


图4-15 4M 线性工作功耗

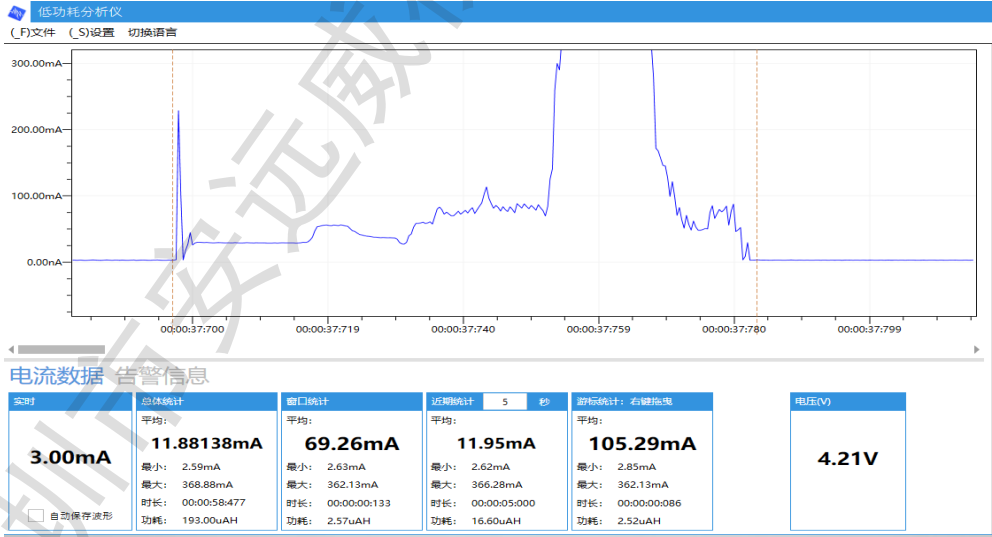


图4-16 4M 线性待机功耗



图4-17 4M 线性 AOV 周期功耗

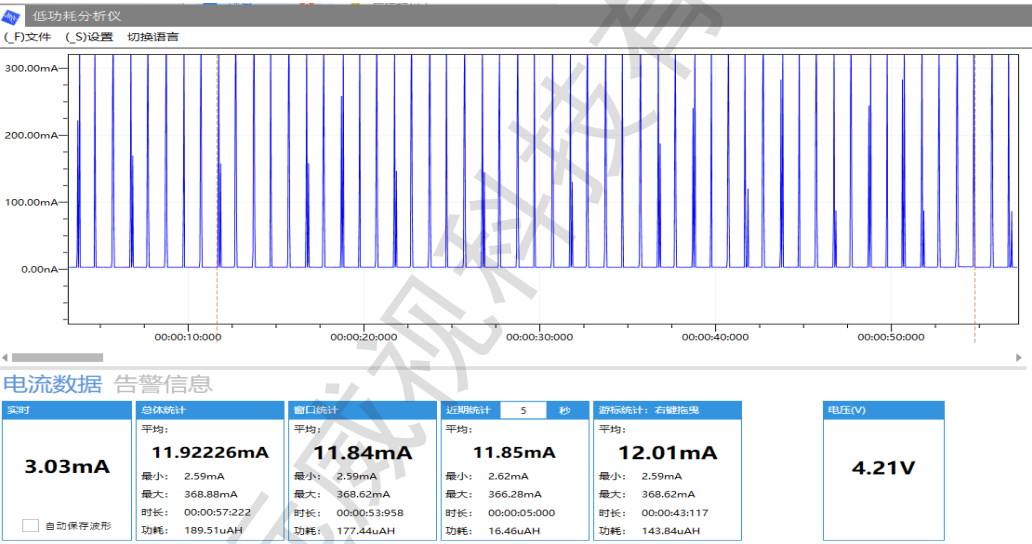


表4-3 4M 线性 AOV 平均工作功耗

工作平均功耗 (mW)	工作周期 (ms)	待机平均功耗 (mW)	待机周期 (ms)	整机平均功耗 (mW)
443.2	86	12.76	912	49.77784
			2912	25.09928
			14912	15.227856

4.3.4 4M AINR

图4-18 4M AINR proc 曝光时间

-----ISP PIPE[0] AE INFO-----						
Version: [AE V1.1], Build Time[Oct 31 2025 16:37:39]						
sys_gain	int_time_sf	int_time	exp_value	ori_avg	exp_ratio	
2448525	0	83143	603099143	1395	0	
ae_target	speed	fps	real_fps	black_delay	white_delay	
7168	128	12.00	12.00	8	0	
wdr_mode	anti_flick	slow_mode	gain_thr	manual_exp_en	manual_exp_val	
0	0	1	0	0	1024000	

图4-19 4M AINR 工作功耗

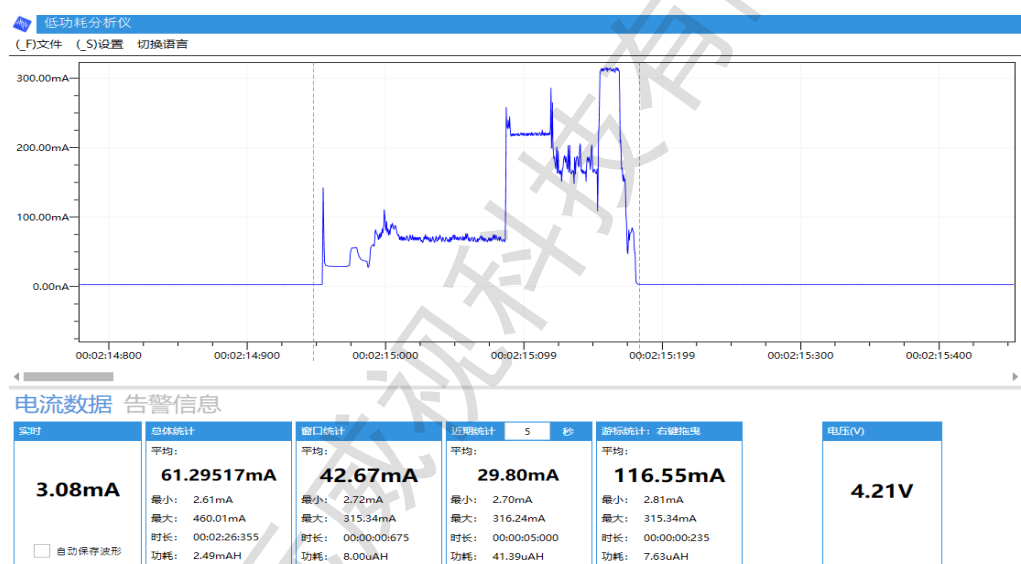


图4-20 4M AINR 待机功耗

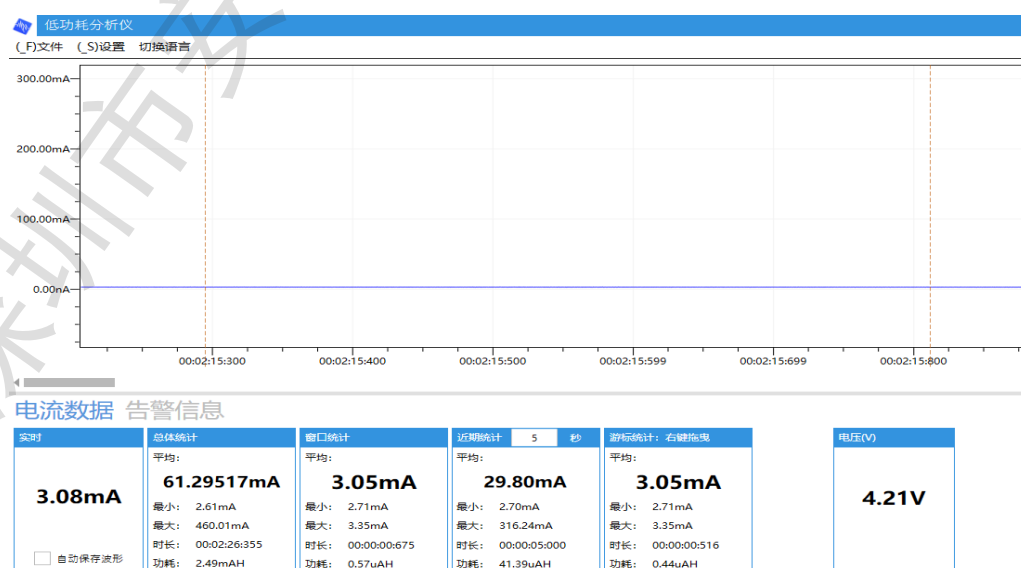


图4-21 4MAINRAOV 周期功耗

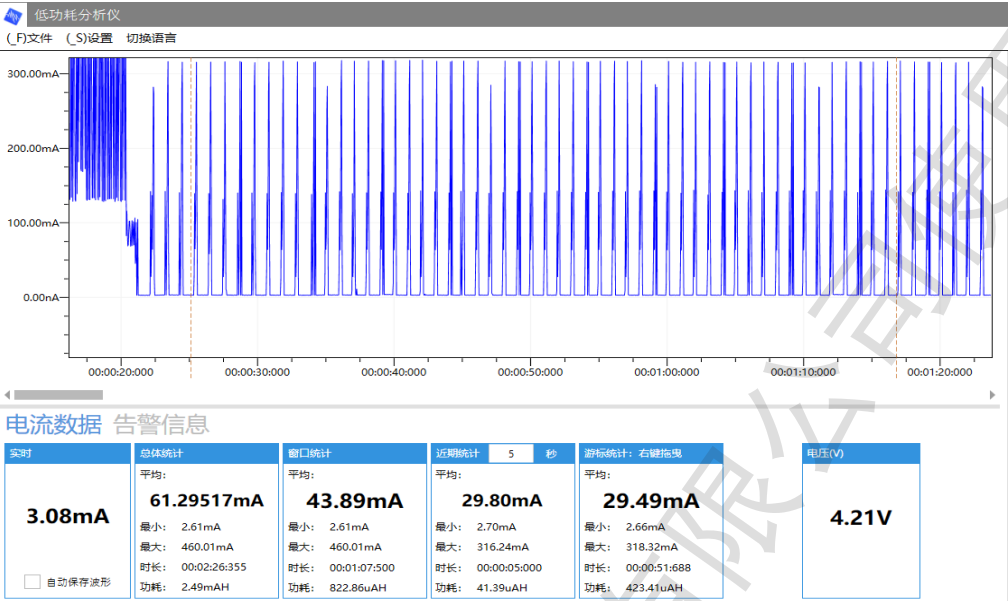


表4-4 4MAINRAOV 平均工作功耗

工作平均功耗 (mW)	工作周期 (ms)	待机平均功耗 (mW)	待机周期 (ms)	整机平均功耗 (mW)
490	235	13.3	765	125.3245
			2765	50.64
			14765	20.76

5.1 sensor 数据篇

5.1.1 待机唤醒后出现 isp fe end/ vpss 获取帧失败

isp fe end / vpss 获取失败，依次排查：

- 需排查 sensor 是否有适配 aov。
- 排查当前 sensor 是否需要 mclk 常供。
- sensor reset 是否在待机过程中被拉低。

5.1.2 AOV 录制数据出现裂屏等问题

- sensor reset 是否在待机过程中被拉低。
- 排查当前 sensor 是否需要 mclk 常供。
- 确认当前 sensor 序列帧率和 AOV 配置帧率，建议适当降低帧率，增大 sensor 消影区。

5.1.3 SENSOR1 由内核驱动控制，SENSOR0 由用户控制

- 内核驱动控制：
sample_aov.c 中 [sample_comm_aov_misc_init](#) 初始化 SDK 驱动控制 sensor，用户可参考实现。
- 用户控制：
sensor 进入睡眠模式：sample_aov_sensor_standby
sensor 退出睡眠模式：sample_aov_sensor_resume(用户可在 xmedia_pm_suspend 后调用，确保 app resume 第一时间恢复 sensor)

5.2 工作周期篇

5.2.1 AOV 出现工作周期不匹配问题

- a. 确认各阶段打印是否关闭
- b. 确认 flash 是否提频
- c. 确认是否关闭串口自举（关闭后，无法通过串口烧录）
- d. 确认 i2c 频率是否有相应调整

5.3 其他篇

5.3.1 AOV 出现待机无法唤醒

- a. 确认整体应用流程耗时是否超过 RTC 周期耗时。
- b. a 条件满足，判断是否有应该卸载的驱动未卸载。