



Sensor 调试指南

文档版本 03
发布日期 2018-07-10

版权所有 © 深圳市海思半导体有限公司 2017-2018。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前 言

概述

本文为对接不同 Sensor 的程序员而写，目的是供您在进行 Sensor 对接的过程中提供对接步骤及注意事项的参考。该指南主要包括一款新 Sensor 对接的驱动开发流程、新 Sensor 在 SDK 中的适配等。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3516C	V300
Hi3516E	V100
Hi3519	V101
Hi3516A	V200
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3516C	V500
Hi3516D	V300

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。

文档版本 03 (2018-07-10)

第 3 次版本发布

添加 Hi3519AV100 相关内容

文档版本 02 (2017-12-20)

第 2 次正式版本发布。

1.4 小节涉及修改

文档版本 01 (2017-08-24)

第 1 次正式版本发布。



目 录

1 Sensor 调试指南	1
1.1 调试流程.....	1
1.2 准备材料.....	2
1.2.1 确认主芯片规格.....	2
1.2.2 Sensor datasheet.....	2
1.2.3 Initialize Settings	2
1.3 采集图像.....	2
1.3.1 硬件准备就绪.....	2
1.3.2 完成初始化序列配置.....	2
1.3.3 Sensor 输出.....	3
1.4 ISP 基本功能.....	3
1.5 完成 AE 配置	5
1.6 完善功能.....	6
1.7 颜色、去噪等校正.....	7
1.8 图像质量调优.....	7



插图目录

图 1-1 Sensor 调试流程图.....	1
-------------------------	---



1 Sensor 调试指南

1.1 调试流程

请按照如图 1-1 所示流程进行调试。

图1-1 Sensor 调试流程图





1.2 准备材料

1.2.1 确认主芯片规格

支持 Master 模式，支持的线性、WDR 接口模式，支持输入频率上限。

1.2.2 Sensor datasheet

- 确认图像传输接口模式，输出频率。
- 确认曝光时间、增益如何设置，帧率如何修改。
- 确认在 WDR 模式下的以上两项。
- LVDS 接口，需要确认同步码。

1.2.3 Initialize Settings

获取 Sensor Initialize Settings，一般至少要准备最大规格和标准分辨率两种序列。

1.3 采集图像

1.3.1 硬件准备就绪

首先验证是否可以读写 Sensor 寄存器。

利用 i2c_read/ i2c_write 命令，或 ssp_read/ssp_write 命令，测试 Sensor 寄存器读写。

该命令集成在默认的文件系统中，可直接调用。

1.3.2 完成初始化序列配置

配置初始化序列，建议参考版本发布包里面的 sensor 驱动，以便于快速开发。为了方便调试，此时要排除 AE 配置及帧率配置的干扰。

步骤 1. 准备 Sensor 驱动

- 可以基于一款规格相近 Sensor (master/slave, i2c/spi, wdr/linear) 驱动修改，尝试编译出 Sensor 库。具体可参看 isp/sensor/hi35xx/xxxx 目录下的 xxx_cmos.c 和 xxx_sensor_ctl.c 文件进行修改。
- 修改 cmos_set_image_mode 函数，及 cmos_get_isp_default 中的 u32MaxWidth, u32MaxHeight 参数。使该 sensor 分辨率、帧率可以被正确设置。
- 在加载 ko 的脚本 load35xx 中修改 Sensor 时钟配置、I2C/SPI 接口 pin mux, vi 时钟, isp 时钟等寄存器修改。适配时，可基于相似规格的 Sensor 修改。

步骤 2. Sensor 初始化序列

- 实现 void sensor_init() 函数。参考 sensor 手册或者 sensor 厂家提供的 sensor 序列实现这个函数。
- 在 xxx_sensor_ctl.c 填写 sensor 寄存器的基地址 sensor_i2c_addr, 地址的比特位宽 sensor_addr_byte, 寄存器的比特位宽信息 sensor_data_byte。



- 在 xxx_cmos.c 文件中，注释掉全部 sensor_write_register，并在 cmsos_get_sns_regs_info 函数里，把 u32RegNum 配置为 0。以使 AE 不配置 sensor，排除干扰。

----结束

1.3.3 Sensor 输出

本部分是基于 mpp 目录下的 sample 做整个通路的输出说明。主要在已完成了 sensor 序列的前提下做的。其步骤主要包括：MIPI、VI、ISP 以及 VPSS 的配置。这些配置可以参考已有 sensor 的配置进行简单修改即可。如果已经有集成的环境直接配置参数就可以运行，比如 Hisi PQTool 的启动脚本，对应 sensor 的目录有启动的配置文件，只需要配置正确即可。

- 步骤 1. 在完成初始化的配置之后，可在 ISP 目录下编译即可生成新的 Sensor 的库，新库的路径为 mpp/lib/ libsns_XXX.a 和 mpp/lib/ libsns_XXX.so。
- 步骤 2. 基于 mpp 的 sample 对新 Sensor 进行验证。在 sample/Makefile.param 文件中新增一款 Sensor 的编译配置 SENSOR_TYPE，然后添加对应的 libsns_XXX.a 文件
- 步骤 3. 在 sample_comm.h 中的 SAMPLE_VI_MODE_E 中添加该 sensor 类型，注意和 sample/Makefile.param 文件中新增的 SENSOR_TYPE 一致。然后再 sample_comm_isp.c 中 SAMPLE_COMM_ISP_Init 函数中添加这个 sensor 类型的属性，如：Bayer pattern，帧率，宽高信息。
- 步骤 4. 配置 MIPI 属性，在 sample_comm_vi.c 中 SAMPLE_COMM_VI_SetMipiAttr 添加 MIPI 属性，调试 MIPI/LVDS 部分参考《MIPI 使用指南》。
- 步骤 5. 配置 VI 属性。在 sample_comm_vi.c 中 SAMPLE_COMM_VI_StartDev 添加 VI 属性。
- 步骤 6. 编译并运行相应的应用程序 sample_vio，如果一切顺利，此时整个系统已经运行。可以通过 cat /proc/umap/isp 或者 cat /proc/umap/hi-mipi 等查看信息。
- 步骤 7. 如果 ISP 没有中断，请先检查 Sensor 输入时钟、输出信号及 Sensor 寄存器配置是否正常。具体操作请查阅《Hi35xx 专业型 HD IP Camera SoC 用户指南》或《Hi35xxVxxx ultra-HD Mobile Camera SoC 用户指南》。
- 步骤 8. 若发现 MIPI、VI、ISP 等都正常，并想进行图像质量调节，可以把上述的配置移植到 PQTool 的对应 sensor 配置文件中（在 config 目录下新建一个 sensor 目录，参考类似 sensor 的配置做相应的修改即可），点播看图。

----结束

1.4 ISP 基本功能

本章节涉及 Sensor 部分，请仔细阅读 Sensor 的 Datasheet，或联系 Sensor 原厂 FAE。

结构体说明请参考《HiISP 开发参考》。

驱动文件一般分为 xxx_cmos.c 文件，xxx_cmos_ex.h 和 xxx_sensor_ctl.c 文件，分别用于 ISP 功能和初始化序列，xxx_cmos_ex.h 文件用于存放定义的驱动文件中的全局变量。



驱动文件共有 3 个 callback 函数，是 sensor 驱动向 Firmware 注册函数的接口。
HI_MPI_ISP_SensorRegCallBack(), HI_MPI_AE_SensorRegCallBack(), HI_MPI_AWB_SensorRegCallBack(), 分别对应 ISP、海思 AE 及海思 AWB。

开发流程

ISP 基本功能，请按如下顺序实现：

1. `cmos_set_image_mode()`, `cmos_set_wdr_mode()`。
2. `sensor_global_init()`。
3. `sensor_init()`, `sensor_exit()`。
4. `cmos_get_isp_default()`, `cmos_get_isp_black_level()`。

注意事项

- `cmos_set_image_mode()`
该函数用于区分不同分辨率，用全局变量 `gu8SensorImageMode` 传递分辨率模式。
请注意返回值，返回“0”表示重新配置 Sensor，会调用 `sensor_init()`，返回“-1”表示不用重新配置 Sensor，无动作。
- `cmos_set_wdr_mode()`
该函数用于区分不同 WDR 模式，用全局变量 `genSensorMode` 传递。
请注意 `gu32FullLinesStd` 和 `gu32FullLines` 的区别。`gu32FullLinesStd` 是当前分辨率及 WDR 模式下，标准帧率（一般为 30fps）时的总行数。`gu32FullLines` 是实际总行数，该参数会在其它函数中，由于降帧的原因，基于标准行数 `gu32FullLinesStd` 及帧率修改。
不同 WDR 模式，一般会修改 AE 相关函数，ISP default 内各个参数以及初始化序列。
- `sensor_init()`
请根据不同的分辨率及 WDR 模式配置不同序列。
- `sensor_exit()`
实现参考类似 sensor 的驱动即可。
- `cmos_get_isp_default()`
该函数配置基本是调试或校正参数，可以在调试及校正时修改参数。
请注意不同 WDR 模式参数可能不一样，比如 Gamma，DRC 等。具体请参考《HiISP 开发参考》。
- `cmos_get_isp_black_level()`
在这个函数里面配置 RAW 数据四个通道的黑电平。



注意

有些类型的 sensor 的黑电平会随着 gain 值的变化而漂移，这时需要在不同的 ISO 值下分别校正出对应的黑电平值，在 `cmos_get_isp_black_level()` 函数内进行相应的实现。

- `sensor_global_init()`



该函数配置了 sensor 初始化的相关配置，包括分辨率、WDR 模式、gu32FullLinesStd 的默认值，初始化状态值及其他相关的状态值。

1.5 完成 AE 配置

完成 AE 配置后，图像就基本正常了。

开发流程

AE 配置，请按如下顺序实现：

1. `cmos_get_sns_regs_info()`。
2. `cmos_get_ae_default()`, `cmos_again_calc_table()`, `cmos_dgain_calc_table()`。
3. `cmos_get_inttime_max()`。
4. `cmos_gains_update()`, `cmos_inttime_update()`。
5. `cmos_fps_set()`, `cmos_slow_framerate_set()`。

注意事项

- `cmos_get_sns_regs_info()`
该函数用于配置需要确保同步性的 sensor、ISP 寄存器，如曝光时间、增益及总行数等。虽然这些寄存器可以通过直接调用 `sensor_write_register()` 来配置，但无法保证同步性，可能出现闪烁。所以这些寄存器请一定要用该函数配置。
`u8DelayFrmNum` 是寄存器配置延时。举个例子，很多 Sensor 的增益是下一帧生效，但曝光时间是下下帧生效，所以需要增益晚一帧配置，以使增益和曝光时间同时生效，这时就需要用 Delay 的功能。配置 `u8Cfg2ValidDelayMax` 是控制 ISP 与 sensor 同步，ISP 包括 ISP Dgain 和 WDR 曝光比等参数，可通过检查 ISP Dgain 是否与 sensor gain 同步来检查参数正确性。该参数的意义是生效时间，一般会比最大的 sensor 寄存器延迟多 1。
 - `bUpdate` 用于控制该寄存器是否更新，如果不用修改，可以置为 false。
- `cmos_get_ae_default()`
 - 请根据 sensor 修改参数。`enAccuType` 是计算精度的类型，常用 `AE_ACCURACY_TABLE` 及 `AE_ACCURACY_LINEAR`。而 `AE_ACCURACY_DB` 因为 CPU 计算精度问题，除非精度很低的，均由 `TABLE` 的方式代替。
 - `LINEAR` 方式是指曝光时间或增益以固定步长线性递增。比如每一步增长 0.325 倍，或曝光时间每一步增长 1。步长由 `f32Accuracy` 决定。
 - `TABLE` 方式一般用于增益，指每一步可以达到的增益通过查表的方式，在 `cmos_again_calc_table()` 或 `cmos_dgain_calc_table()` 函数中计算得到。此时 `f32Accuracy` 失去意义，不生效。

海思 AE 默认计算顺序是先分配曝光时间，其次 again，然后 digain，最后 isp dgain。可以通过设置 AE Route 或 AE RouteEx 来调整分配顺序。
- `cmos_again_calc_table()`, `cmos_dgain_calc_table()`



这两个函数输入、输出完全一致，分别对应 Again 和 Dgain 的 TABLE 方式。下面以 Again 为例说明。

- pu32AgainLin 同时做输入和输出。做输入是 AE 计算出来的期望增益，1024 表示 1 倍。在该函数中，要查询到一个 sensor 可以实现的，小于该增益的最大增益。并重新赋给该参数作为向 AE 的输出。
- pu32AgainDb 是输出，AE 内部不用于运算，只是作为函数 `cmos_gains_update()` 的输入。一般用于传递当前增益的 sensor 寄存器值。

例如：某 sensor 增益按 0.3dB 递增。sensor 寄存器值从 0 开始，每增加 1，对应增益分别为 0dB, 0.3dB, 0.6dB, 0.9dB...

离线算出一个将 dB 转化为线性倍数的查找表，为 1024, 1060, 1097, 1136...

在函数中将输入的增益与查找表比对，假如输入为 1082，那查出来可用的最大增益是 1060，返回 1060 为实际生效的增益。

- `cmos_get_inttime_max()`

该函数只在 xto1 WDR 模式下生效，用于计算不同曝光比的时候，曝光时间的最大值。

一般是行合成模式才需要。因为行合成模式，曝光时间的限制为长曝光时间加短曝光时间的和要小于一帧长度。所以不同曝光比下，最大曝光时间有差异，需要重新运算。

- `cmos_gains_update()`, `cmos_inttime_update()`

这两个函数，是根据输入的 Again、Dgain 或曝光时间配置 sensor 寄存器。精度模式采用 TABLE 时，输入参数值为对应

`cmos_again_calc_table()/cmos_dgain_calc_table()` 函数中返回的 pu32AgainDb、pu32DgainDb。

精度模式采用 Linear 时，输入参数为生效的增益、曝光时间除以 f32Accuracy。比如 f32Accuracy 为 0.0078125，实际生效增益为 1.5 倍时，输入值为 $1.5 / 0.0078125 = 192$ 。

Xto1 WDR 模式，需要分别配置长短每一帧的曝光时间。`cmos_inttime_update()` 会被调用 X 次，分别传入不同帧曝光时间，第一次传入短帧。

- `cmos_fps_set()`, `cmos_slow_framerate_set()`

`cmos_fps_set()` 函数为手动帧率配置函数，需要根据传入的帧率配置 sensor 对应的寄存器，实现改变 sensor 帧率的功能，并返回实际生效的帧率及最大曝光行数。

`cmos_slow_framerate_set()` 函数为自动降帧配置函数，需要根据当前曝光实际需要的最大曝光行数配置 sensor 对应的寄存器，实现 sensor 的降帧功能，并返回实际生效的最大曝光行数。

1.6 完善功能

完善所有其它的函数，确保所有功能工作正常。

由于 AE 中的同步性最容易出错，请重点验证同步。



1.7 颜色、去噪等校正

请根据《图像质量调试工具使用指南》校正 sensor 参数。

1.8 图像质量调优

图像质量调优请参阅对应的《ISP 图像调优指南》。