Sensor 调试指南

文档版本　01

发布日期　2016-11-24

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

本文为基于《HiMAPI V1.0 媒体处理开发参考》进行产品开发过程中对接不同 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 MobileCam SDK 中的适配等。

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 01 (2016-11-24)

第 1 次正式版本发布。

目 录

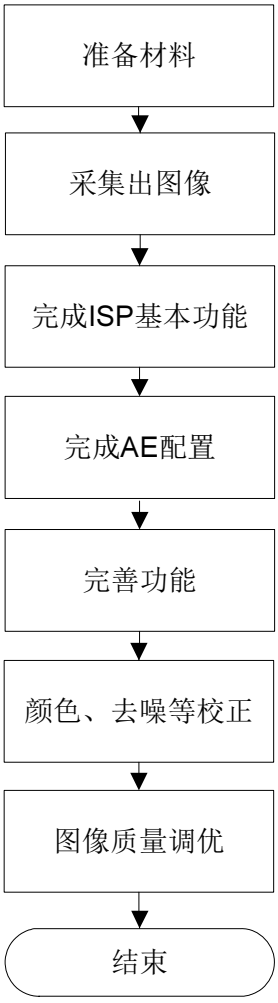
插图目录

1 对接一款新 Sensor

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/Slave 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

- 确认是 Master 模式还是 Slave 模式，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。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 mapi/code/mediaserver/component/sensor/hi3518e/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 senosr 分辨率、帧率可以被正确设置。
- 修改 mapi/release/hi3518e/signal/slave/init/sdk_init.c 中的 Sensor 时钟配置、I2C/SPI 接口 pin mux 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。
- 在 xxx_cmos.c 文件中，注释掉全部 sensor_write_register，并在 cmos_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

步骤 1. 在完成初始化的配置之后，可在 mapi 目录下编译即可生成新的 Sensor 的库，新库的路径为 mapi/code/mediaserver/component/sensor/hi3518e/xxx/libsns_xxx.a。

步骤 2. 参看 2 “新 Sensor 在 MobileCam SDK 中适配”中新 Sensor 在 MobileCam SDK 中的适配举例，添加相应 Sensor 的配置，在 mapi/code/mediaserver/configs/sensor/hi35xx/目录下新增一款 Sensor 的配置。

步骤 3. 在 mapi/release/hi3518e/signal/slave/init/sdk_init.c 中增加新的 Sensor 类型，并增加新 Sensor 时钟配置、I2C/SPI 接口 pin mux 等寄存器配置。

步骤 4. 基于 MobileCam SDK 的 sample 对新 Sensor 进行验证。在 mapi/sample/hi3518e/HuaweiLite.param 文件中新增一款 Sensor 的编译配置，包括对 libsns_cfg_xxx.a、libsns_xxx.a、I2C/SPI 接口库、等库的链接。

步骤 5. 为 Huawei LiteOS 端的修改，用户编译应用完成之后需要将 Huawei LiteOS 端生成的 mapi/sample/hi3518e/xxx/HuaweiLite/xxx.bin 文件烧写到单板中，新适配的 Sensor 才能生效，如 mapi/sample/hi3518e/venc/HuaweiLite/sample_venc.bin。

步骤 6. 运行相应的应用程序 mapi/sample/hi3518e/xxx/HuaweiLite/xxx.bin，如果一切顺利，此时已可以看到图像。

步骤 7. 如果在运行起来在 Huawei LiteOS 端报”not support sensor mode”的错误，则说明 Sensor 模式配置得不对，请按照 2 “新 Sensor 在 MobileCam SDK 中适配”的配置说明的注释部分认真检查配置。

步骤 8. 如果看不到图像，请先检查 Sensor 输入时钟、输出信号是否正常，检查 Sensor 寄存器配置是否正常。

步骤 9. 目前不支持并口 Sensor 的对接，如有需求，请联系海思 FAE。

步骤 10. 最后检查图像是否正常，若不正常，可检查 sensor_interface_cfg_params.c 配置文件中的 Bayer 格式、比特位掩码、MIPI 裁剪等是否正确。

----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。

驱动文件一般分为 xxx_cmos.c 文件和 xxx_sensor_ctl.c 文件，分别用于 ISP 功能和初始化序列。slave 模式 sensor 还有 xxx_slave_priv.c 文件，用于存储行场同步信号参数。

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。slave 模式 sensor 的行场同步信号信息也要在该函数中初始化。

- 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，尤其是 Gamma，及有的芯片需要配置 GammaFE。

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。
  - stSlvSync 用于控制 slave 模式的 sensor 的行场同步信号。

- 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。

2 新 Sensor 在 MobileCam SDK 中适配

需要适配的配置文件在 mapi/code/mediaserver/configs/sensor/hi35xx/xxx/ sensor_interface_cfg_params.c，其结构的定义为 mapi/code/mediaserver/configs/sensor/ include/ sensor_interface_cfg_params.h。

2.1 配置举例说明

2.1.1 Sensor 的配置举例

```
/**<MIPI/LVDS接口时序配置,每增加一种时序配置user_mipi_dev_attr的序号需要增加一项，否则可能造成运行起来找不到对应的时序。可看到以下配置中有两种时序*/
const mipi_dev_attr_t user_mipi_dev_attr[2] =
{
   /*mipi  mode 0*/
   {
    .raw_data_type = RAW_DATA_12BIT,
    .lane_id =  {0, -1, -1, -1, -1, -1, -1, -1}
   }
    /**<可以在此增加新的MIPI/LVDS接口时序*/
};
/**<Sensor时序配置，每增加一种Sensor时序配置，user_mipi_intf的序号需要增加一项，否则可能造成运行起来找不到对应的时序*/
HI_SENSOR_MIPI_LVDS_INTF_S user_mipi_intf[2] =
{

    {
        .phy_clk_share = HI_PHY_CLK_SHARE_PHY0,
        .stPhyCmv =
        {
            .bSetPhyCmv = HI_FALSE,
            .enPhyCmv = PHY_CMV_BUTT
```

```
        },

        /**< img_rect对应mipi/lvds的裁剪属性，在对接Sensor时，如果发现输出的图像有黑边时，通常需要调节此处的x、y值，裁剪黑边时保证按需裁剪，如果裁剪过多超出范围可能会造成mipi/lvds输出无数据。*/
        .img_rect =
        {
            .x = 0,
            .y = 0,
            .width = 1280,
            .height = 720
        },
        .ptr_dev_attr =(void*)&user_mipi_dev_attr[0],
        .au32CompMask = {0xfff00000,0},

    /**<可以在增加新的Sensor时序配置。*/
};
/**< sensor_mipi_intf_cfg表示所有的Sensor时序配置总结构，在配置模式时sensor_mode，需要选择对应模式的Sensor时序的序号，该序号为user_mipi_intf[]数组的下标。若增加了一项时序，需将dev_attr_total_num增加1。否则，可能会找不到对应的时序。*/
HI_SENSOR_MIPI_LVDS_INTF_CFG_S sensor_mipi_intf_cfg =
{
    .dev_attr_total_num = 1,
    .psensorIntf = (void*) user_mipi_intf
};

/**< sensor_mode表示Sensor模式配置，并需要配置该模式下的Sensor时序，时序序号s32SensorIntfSeqNo为user_mipi_intf数组下标，模式中的宽、高属性必须4字节对齐。HI_MAPI_Sensor_GetAllModes(见《HiMAPI V1.0 媒体处理开发参考》的“视频采集”章节中介绍)获取的对应Sensor的所有模式如下列表，HI_MAPI_Sensor_SetAttr(见《HiMAPI V1.0 媒体处理开发参考》的“视频采集”章节中介绍)接口中传入的模式也必须是如下列表中的其中一个，否则会提示模式不存在。*/
const HI_SENSOR_MODE_S sensor_mode[7] =
{
    /*4K@30*/
    {
        .s32Width = 3840,
        .s32Height = 2160,
        .s32FrameRate = 30,
        .s32SensorIntfSeqNo = 0,
        .enWdrMode = HI_MPP_WDR_MODE_NONE
    },
```

```
/*4K@60*/
{
    .s32Width = 3840,
    .s32Height = 2160,
    .s32FrameRate = 60,
    .s32SensorIntfSeqNo = 0,
    .enWdrMode = HI_MPP_WDR_MODE_NONE
},
/*1080p@60*/
{
    .s32Width = 1920,
    .s32Height = 1080,
    .s32FrameRate = 60,
    .s32SensorIntfSeqNo = 1,
    .enWdrMode = HI_MPP_WDR_MODE_NONE
},
/*1080p@120*/
{
    .s32Width = 1920,
    .s32Height = 1080,
    .s32FrameRate = 120,
    .s32SensorIntfSeqNo = 1
},
/*720p@240*/
{
    .s32Width = 1280,
    .s32Height = 720,
    .s32FrameRate = 240,
    .s32SensorIntfSeqNo = 2,
    .enWdrMode = HI_MPP_WDR_MODE_NONE
},
/*720p@30*/
{
    .s32Width = 1280,
    .s32Height = 720,
    .s32FrameRate = 30,
    .s32SensorIntfSeqNo = 0,
    .enWdrMode = HI_MPP_WDR_MODE_NONE
},
/*12M*/
{
    .s32Width = 4000,
    .s32Height = 3000,
    .s32FrameRate = 28,
```

```
        .s32SensorIntfSeqNo = 3,
        .enWdrMode = HI_MPP_WDR_MODE_NONE
    }
};

/**< user_dev_attrs表示所有Sensor配置的数组*/
const HI_COMBO_DEV_ATTR_S user_dev_attrs[1] =
{
    /*config SENSOR 0*/
    {
        .stSensorInputAttr =
        {
            /**< Bayer格式*/
            .enBayerFormat = HI_BAYER_BGGR,
            .s32start_x = 0,
            .s32start_y = 0
        },
        .stSensorIntf =
        {
            /**< bInitByFastboot为快速启动预留接口*/
            .bInitByFastboot = HI_FALSE,
            /**<Sensor设备号，序号由0递增，若为2个Sensor，则依次为0,1*/
            .devno = 0,
            /**< input_mode表示Sensor与芯片的数据信号接口类型,
INPUT_MODE_MIPI/ INPUT_MODE_LVDS*/
            .input_mode = INPUT_MODE_MIPI,
            /**< Sensor与芯片的控制信号接口类型,
HI_SENSOR_COMMBUS_TYPE_I2C/HI_SENSOR_COMMBUS_TYPE_SPI*/
            .enSensorCommBusType = HI_SENSOR_COMMBUS_TYPE_I2C,
            .pstIntf= (void*)&sensor_mipi_intf_cfg
        },
        /**<Sensor 0的所有模式*/
        .stSensorMode =
        {
            /**<在添加模式之后也需要对s32SensorModeNum进行修改，否则会造成找不到所
添加的模式/
            .s32SensorModeNum = 7,
            .pstSensorMode = (void*) sensor_mode
        },
    }
    /**<在此可以配置另外一个Sensor，举例可见3.2.2.2章节的双Sensor的配置举例*/
```

```
};
/**<Sensor的总配置*/
extern const HI_SENSOR_CFG_S sensor_cfg =
{
    /**<如果新增一个Sensor，则s32TotalSensorNum需要增加1*/
    .s32TotalSensorNum = 1,
    .pstSensors = (void*) user_dev_attrs
};
```

以上为 MIPI 接口的配置举例，LVDS 的与 MIPI 类似，具体可参看发布目录：

mapi/code/mediaserver/configs/sensor/ hi3518e / ov9732 /sensor_interface_cfg_params.c，该文件为 MIPI 接口配置。