Huawei LiteOS 快速启动应用指南

文档版本　01

发布日期　2017-06-12

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

本文档主要介绍基于 Huawei LiteOS 的快速启动优化措施。大部分优化措施对于所有芯片都是适用的，如果有不适用的情况，文中会具体描述。

说明

本文描述以 Hi3516A 为例。未有特殊说明，Hi3518EV20X、Hi3516CV200 与 Hi3516AV100 一致。

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
Hi3516A	V100
Hi3516D	V100
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200

读者对象

本文档（本指南）主要适用于：

- 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 (2017-06-12)

2.2.1 小节，表 2-1 涉及修改

2.6 小节涉及修改

3.4.3 小节，表 3-3 涉及修改

文档版本 00B01 (2016-05-10)

第 1 次临时版本发布。

目 录

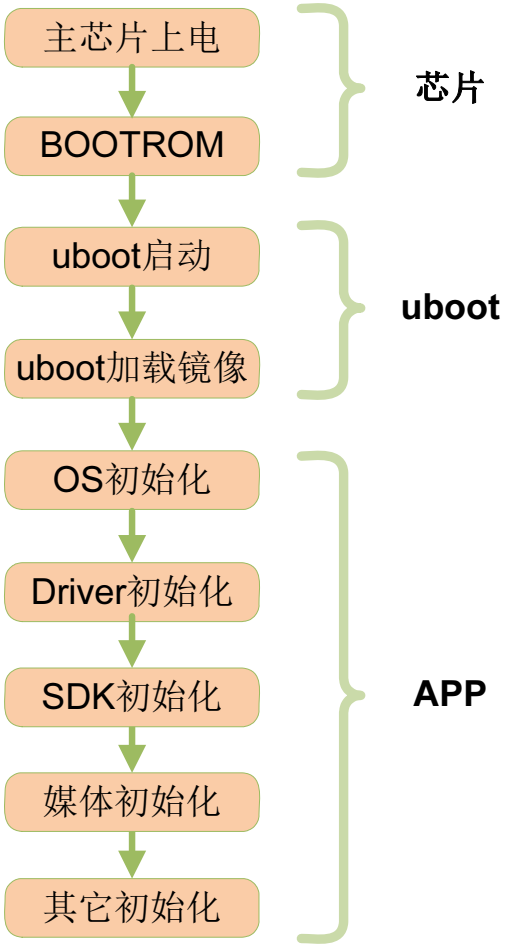
插图目录

表格目录

1 快速启动综述

对于 DV 类产品，启动时间的长短直接影响用户体验、电池使用时长，因此缩短启动时间是一个非常关键的技术点，供参考的 DV 产品的启动流程如图 1-1 所示。

图1-1 通用 DV 启动流程



本文基于 Huawei LiteOS 来介绍现已经经过验证的快速启动优化措施（其中大部分优化措施也是可以应用到 Linux 平台上的），优化后的启动流程总体上仍然和图 1-1 保持一致，但在启动速度上，可以达到百毫秒级，快速启动用到的优化选项详见表 1-1。

表1-1 快速启动优化措施汇总表

序号	阶段	优化措施	收益
1	芯片	从硬件上关闭 BOOTROM 启动	提升约 50ms
2	uboot	裁剪 uboot(Miniuboot)，加快 uboot 启动流程	提升约 50ms
3	uboot	使用压缩、解压技术，减少镜像加载时间，取决于芯片是否支持硬件解压功能。	依赖于镜像大小，一般 4MB App，使用压缩，可以提升约 30ms
4	uboot	Flash 驱动优化	APP 镜像越大，收益越明显示，目前使用 mx25l25635f 等器件已达到 32MB/S 的读速
5	APP	使用分散加载特性，优先加载媒体模块	根据具体 APP 决定，收益一般在 30ms-50ms 左右
6	APP	将 Sensor 初始化提前到 uboot 中进行	根据 Sensor 决定，收益一般在 2 帧视频时间
7	APP	APP 镜像小型化，缩短 APP 镜像加载时间	使用 APP 镜像大小乘以 Flash 读速即可得知优化收益
8	uboot&APP	关闭 Huawei LiteOS 打印输出（printf/dprintf），如果没有对应的源代码，则忽略这一步	开机时的打印对启动速度影响较大，关闭打印能带来直接的收益
9	APP	优先启动关键模块，延缓非关键模块的初始化	根据具体业务而定
10	APP	对 bss 段内存不需要进行 memset 为 0 的动作	根据内存大小而定，越大块的 bss 内存越明显
11	uboot&APP	将 I2C 频率从 100KHZ 提升到 400KHZ	对于使用 I2C 与 Sensor 进行通讯的，约提升 20ms 以上

部分快速启动措施是一个双刃剑，在带来启动性能提升的同时，会在器件兼容性、单板可调试性、代码可读性等方面带来负作用，因此本文的措施仅供参考，请在实际产品开发过程中根据需要选用。

2 快速启动优化措施

2.1 关闭 BOOTROM 启动

必须使用到 BOOTROM 的场景：

a. 对于 EMMC 器件，开机启动时需要借助 BOOTROM 才能启动

b. 需要空片串口烧录（如 uboot），必须借助 BOOTROM

对于 SPI NOR Flash/SPI NAND Flash 器件，可以开机从 Flash 启动而不需要 BOOTROM，因此在硬件设计上可以考虑关闭掉 BOOTROM，在生产出厂时关闭 BOOTROM 功能，提升 50ms 的开机性能。

说明

- 关于 BOOTROM 的介绍，请参考具体芯片手册，例如《Hi3516A / Hi3516D 专业型 HD IP Camera Soc 用户指南.pdf》。
- 硬件上的设计请参考具体硬件设计用户指南，例如《Hi3516A / Hi3516D 硬件设计 用户指南.pdf》

2.2 使用 Miniuboot

发布包中的标准 uboot 兼容性好、规格齐全、调试方便，因此编译出来的镜像也会比较大，启动速度稍慢。为提升启动速度，可以针对单板量身打造一个 Miniuboot，把调试手段裁剪掉，只留下匹配当前单板器件的 Flash 驱动，能够从 uboot 把 App 镜像正常加载并启动即可。

本小节主要针对裁剪后的流程做出说明。

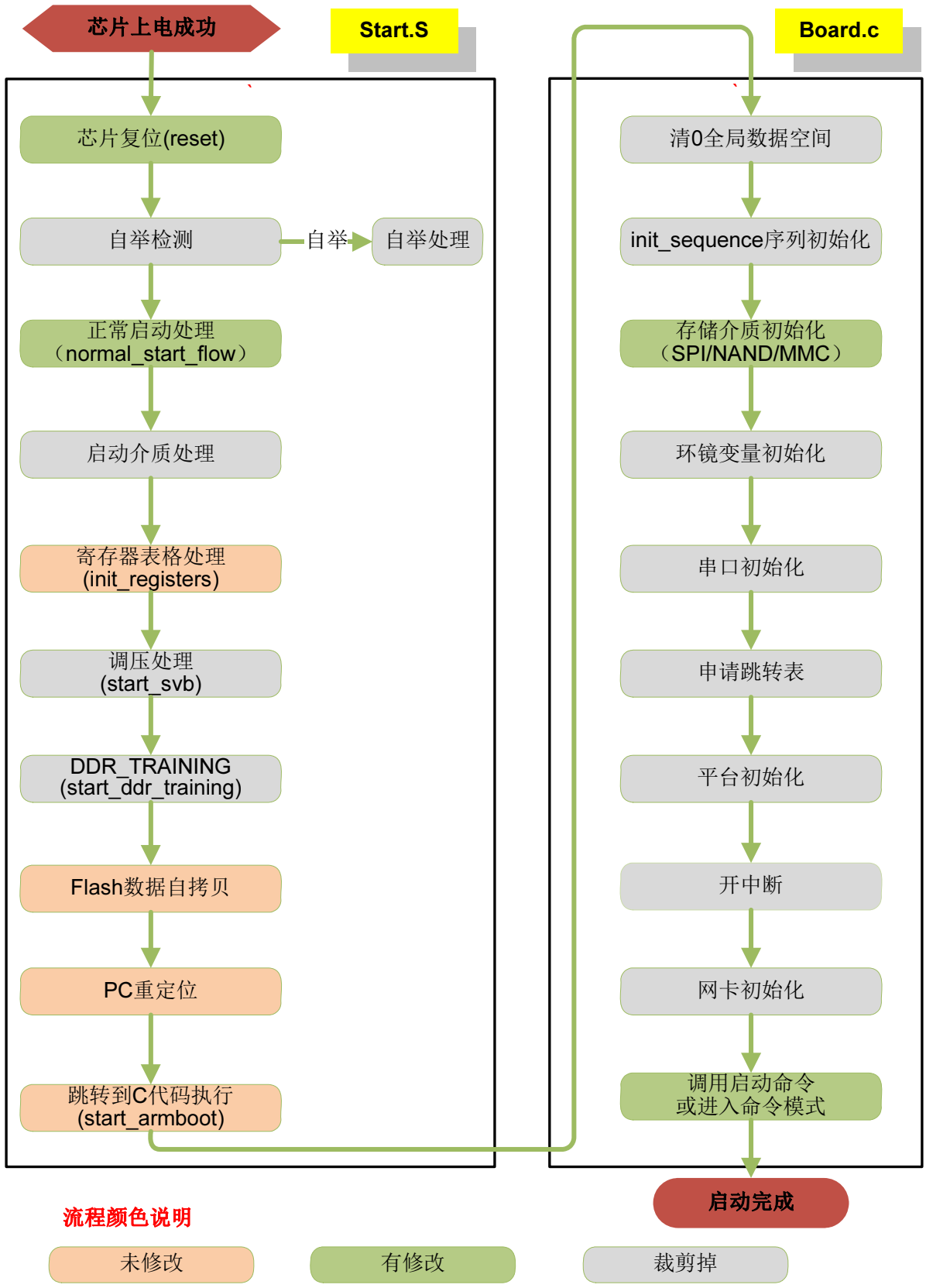
2.2.1 Uboot 裁剪流程说明

表2-1 Uboot 裁剪流程说明

流程	措施	裁剪说明
自举检测	删除	在不需要进行串口烧录的情况下，此部分代码可以裁剪掉

流程	措施	裁剪说明
启动介质处理(SPI/NAND/EMMC)	删除	根据单板上的 Flash 直接处理即可，可以不考虑多 Flash 器件兼容性
调压处理	删除	Hi3518EV200 不支持动态调压，可以删除。
环镜变量处理	删除	Uboot 中不支持环镜变量，直接把 Flash 地址在 uboot 代码中写死
网口初始化	删除	Uboot 不支持网口
存储介质初始化	修改	只初始化当前单板上的 Flash 即可

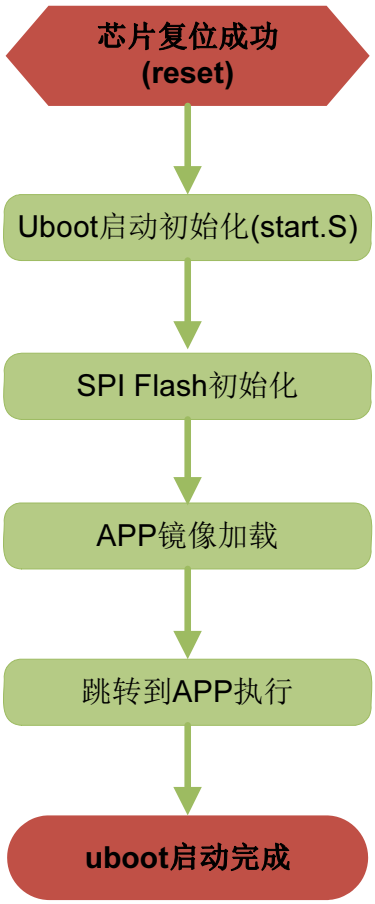
图2-1 标准 U-boot 启动流程及裁剪说明



上述的图 2-1 是一个标准 uboot 启动过程及裁剪概览。表 2-1 是对关键裁剪流程的一个说明。

2.2.2 Miniuboot 启动流程

图2-2 Miniuboot 启动流程



实际的产品化 uboot 可以基于发布的 Miniuboot 包进行增删。

2.2.3 Uboot 裁剪收益

分类	裁剪前	裁剪后	收益	说明
Flash Size	约 188KB	约 25KB	减少 163KB 左右	参考值，实际 uboot 的 size 会根据规格不同而产生偏差
启动速度	324ms	约 8ms	缩短 316ms 左右	Uboot 启动到镜像加载前的参考时间，实际 uboot 的启动时间根据规格不同而产生偏差

2.2.4 如何测量系统启动时间

测量原理

芯片从上电到启动到 uboot 的 start.S 文件中 normal_start_flow 这个点的时间使用示波器测量为 146ms 左右，如果带了 BOOTROM，需要再加 50ms，由于这个时间点不能用软件测量，并且是一个固定值，因此我们只需要知道这个值大小即可，有兴趣的话可以自己使用示波器测试。在 normal_start_flow 初始化 timer2 硬件定时器，并且在 OS 启动过程中不再初始化此定时器，这样就可以从 normal_start_flow 这个点精确测量启动后的时间。

测量步骤

步骤 1. 在 uboot 的 start.S 中添加 timer_init 函数的调用，并屏蔽 init_sequence 数组中的 timer_init

```
normal_start_flow:
        /* timer init */
        ldr     sp, =STACK_TRAINING
        bl      timer_init          /* timer init */
```

步骤 2. 在 timer_init 中添加 timer2 硬件定时器的初始化

步骤 3. 在 Huawei LiteOS 中屏蔽对硬件定时器的初始化（见 hal_clock_initialize_start 函数，如果是不带源码库的 Huawei LiteOS 版本，则无须关心这一步）

步骤 4. 在 uboot 中需要打印时间的地方添加如下代码即可打印实际的时间：

```
{
        unsigned long time_0 = READ_TIMER;
        time_0 = 0xffffffff - time_0;
        time_0 = time_0 * 256;
        time_0 = time_0 / 50000;
        printf("---xxxx clock : %d ms\n", time_0);
    }
```

步骤 5. 在 APP 通过 hi_getmsclock()函数可以获取 TIMER2 的定时器时间：

```
UINT32 hi_getmsclock(void);
```

----结束

说明

标准发布 uboot 的使用，请详见《Hi35xx U-boot 移植应用开发指南》。

2.3 使用压缩、解压技术

Hi3518EV20X 支持 LZMA（GZIP）格式的硬解，因此可以对 APP 镜像进行压缩，在 uboot 中进行解压。Hi3516A 不支持硬件解压。

a. uboot 开发指南：硬解的封装文件为 hi_decompress.c/hi_decompress.h 文件，上层调用样例：

```
void app_hw_dec(unsigned char *dst, int dstlen,
    unsigned char *src, int srclen)
{
    hw_dec_init();
    hw_dec_decompress(dst, &dstlen, src, srclen, 0);
    hw_dec_uinit();
}
```

注意事项：

a) dstlen 不能传空，且要比实际解压后的长度大一些，比如 0x1000，否则会解压出错。

b) dst、src 地址必须 16 字节对齐，否则会解压出错。

c) 注意 GZIP 和 IVE 是复用功能，解压前如果 IVE 已经初始化，则现在先切换成 GZIP，解压完成后，再切换成 IVE。否则会出现解压超时错误。

b. 压缩、解压参考数据

a) 对 4.25MB 的 APP 采用 LZMA 压缩后变为 2.54MB，压缩比为 59.8%

b) SPI Flash 按照 32MB/S 的读速度读取 4.25MB Flash，需要 132ms

c) SPI Flash 按照 32MB/S 的读速度读取 2.54MB，需要 79ms

d) 使用 LZMA 硬解压 2.54MB 数据的时间为 21ms

e) 时间收益：132-(79+21) = 32ms

f) Flash 空间收益：4.25-2.54=1.71MB

结论：使用压缩、解压技术还是能带来可观的收益（启动时间、Flash 空间），Flash 速度越慢收益越大，Flash 镜像越大收益越大。

说明

Hi3518EV20X 所支持的 LZMA 算法与标准的 LZMA 算法不完全一样，已经进行了部分修改，如有需要，需使用发布包中的 LZMA 代码。

2.4 读 Flash 性能优化

在 Huawei LiteOS 系统中，从 Flash 中读取镜像所占的时间一般情况下会占到整个启动时间（不含网络）的 1/4 到 1/6 左右。因此对 Flash 读速率的优化至关重要。这里以 SPI Flash 为例进行说明。

步骤 1. 选用高速 SPI Flash 器件（带 4 线 SPI Flash 读写性能，可向 Flash 供应商咨询或实测）。SPI Flash 建议选择工作时钟频率选择大于等于 100MHz 的 Flash，这样读取速度会更快。

步骤 2. 在 Flash 驱动中关闭 CONFIG_CLOSE_SPI_8PIN_4IO 宏，使 4 线 SPI Flash 读写打开。（注：标准 uboot/Huawei LiteOS 发布包中，为了考虑器件兼容性，此宏是默认打开的）。

步骤 3. 优化 SPI Flash 驱动初始化时间。可以把 uboot 中 hifmc_spi_nor_ids.c 中与当前单板 Flash 器件无关的其它型号全部删除。

步骤 4. Flash 读性能参考数据（基于 winbond w25q64fw）

- 在打开了 CONFIG_CLOSE_SPI_8PIN_4IO 宏的情况下，读速度约 19.6MB/S，读取 4MB 数据需要 204ms。
- 在关闭了 CONFIG_CLOSE_SPI_8PIN_4IO 宏的情况下，读速度约 32MB/S，读取 4MB 数据需要 125ms。

----结束

结论：请根据单板上的 SPI Flash 器件进行具体的代码优化。

2.5 使用分散加载特性

Huawei LiteOS 自带分散加载属性（关于分散加载的说明请阅读《Huawei LiteOS Kernel Deverloper Guide(zh).pdf》），可以将其应用到产品开发中。

使用场景：开机后需要快速启动某关键业务(如开机需要快速抓拍到第一张照片)

快速获取第一帧方法举例：

步骤 1. 在分散第一段镜像中只初始化必要的媒体通路，即把 SENSOR->VI->VPSS->VENC 的通路打通，尽快获取到第一帧，提前获取到的音视频数据缓存在内存中。

步骤 2. 在分散第二段镜像中初始化其它的模块，如网络、SD 卡、录像、抓拍等模块。某些场景下是要抓拍第一张的，可以直接从之前缓存在内存中的音视频 Buffer 中获取。

----结束

2.6 Sensor 初始化提前到 uboot 中进行

Sensor 是一个比较独立的模块，只需要正确初始化 Sensor 序列，Sensor 就可以开始工作，有些 Sensor 初始化后的前几帧可能是黑帧，因此 Sensor 越早初始化越好。

本小节基于 Miniuboot《mini-uboot.tgz》、APP(Sample)、Hi3518EV2DVS_VER_B 硬件单板来介绍如何把 APP 中的 Sensor 初始化移植到 Uboot 中，启动流程图如图 2-3 所示。

a. uboot 中关键修改点：

- 增加 i2c0 初始化(i2c_init())

- 增加 mipi 接口初始化(mipi_init())
- 增加管脚复用、时钟等寄存器初始化(SYS_cfg())
- 增加 sensor 及 sensor 序列初始化(insert_sns() & sensor_init())

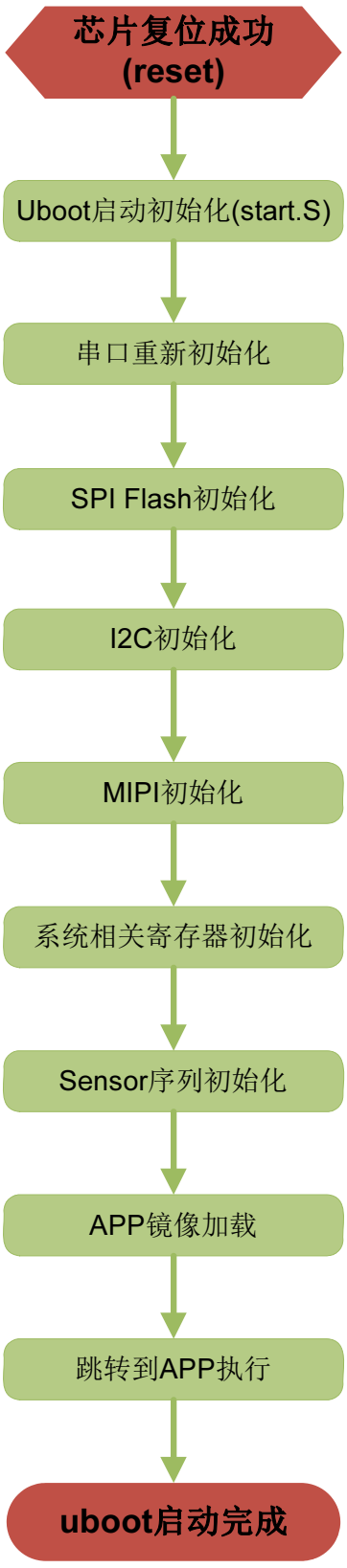
b. SDK 关键修改点：

- Mipi.c 代码中清空 mipi_set_combo_dev_attr 函数，直接返回 HI_SUCCESS 即可
- 清空 Sensor 的初始化函数 sensor_init，直接返回即可

c. APP 关键修改点：

sdk_init.c 代码中管脚复用、时钟、sensor 设置相关寄存器请屏蔽掉

图2-3 Sensor 提前到 uboot 中执行(基于 miniuboot)



2.7 APP 镜像小型化

表 2-2 介绍一些通用的 APP 裁剪方法供参考，这些裁剪措施中有一些（如去调试功能）带有负作用，请根据实际情况选用，或者只在正式发布版本中使用。

表2-2 APP 镜像小型化措施分类

裁剪方法	实施措施	裁剪建议
去调试功能	可以区分为 release 版和 debug 版。 Release 版本下完全不带打印，不带 Proc 信息，任何的调试手段（shell、telnet、nfs、proc 等），但在 Debug 版本下，是有完整的调试信息的，程序可以动态开关	建议正式版本去调试功能，APP 上去调试功能可以带来约 1.1 MB 的收益。
模块解耦	模块间及模块内进行解耦，把最终的调用权利交给 APP 层来决定，这样按 Section 编译后并最终链接时，就会去掉无关的模块代码 具体的实现方案包括：采用注册、回调方式来解耦	如需要落实，需要从设计层面着手
代码精简	1. 去掉冗余代码 2. 寻找替代算法 3. data 段数据变成 bss 段数据（Huawei LiteOS 开机时会把 bss 段清空成 0，因此没必要赋空值的就不需要做这一步）	建议分析镜像 map 文件，对占用 size 较大的模块进行分析裁剪
规格裁剪	根据单板实际需求来增删规格，去掉不需要的模块功能。 如中文字库比较大，对于非中文用户是不需要的	从系统层面进行分析裁剪
镜像分散加载	在 Huawei LiteOS 上，可以分场景将 APP 镜像分成多段，启动后分场景加载	根据实际产品需求进行
去文件系统	由于 Linux 系统本身特性，APP 中很多参数配置都以文件存储在文件系统中。但在 Huawei LiteOS 上，Flash 上的文件系统不是必须的（录制所需的 FAT 文件系统除外），我们可以将参数配置直接存储为裸数据，并删除掉相对应的文件系统（如 yaffs/jffs）	建议去掉文件系统

2.8 关闭打印

经统计，平均每 11 个字符打印就会占用到 1ms 的时间，而在实际系统中，考虑到并发调用、函数封装和参数传递等原因，效率会更低，因此大量的打印所占用的时间非常可观。在不方便从镜像中去掉打印的情况下，可以考虑直接关闭 Huawei LiteOS 中的 printf/dprintf 函数（即直接清空 printf/dprintf 中的实现），使其不能输出打印信息，提升启动速度。

经过实测，在参考设计（sample）中采用此方法后，对于开机第一帧视频的获取能缩短 50ms 以上。

2.9 优先启动关键模块

对于 DV 类产品，通常需要快速的获取到音视频数据并缓存在内存中，而网络、录像、抓拍等模块是可以延后初始化，所需的数据可以从内存缓存中获取。

2.10 Bss 段内存不需要清 0

在 Linux 系统上，对 bss 段中的变量进行清 0（memset）是一个常识。但 Huawei LiteOS 系统中，OS 会在启动时一次性对 bss 段进行清 0，因此没必要再在启动时对 bss 段进行清 0 动作。由于对内存清 0 的时间与内存大小成正比，因此对于一些小尺寸的内存可以不用优化。建议对 1KB 以上的 bss 段内存进行优化即可。

说明

如何知道 bss 段内存的分布？可以在链接生成 APP 镜像时使用 -Map=xxx.map 来生成 map 文件，然后使用文本工具打开分析，里面有各函数、变量所占 size 的统计。

2.11 I²C 提频到 400KHZ

Huawei LiteOS 考虑到器件兼容性，在 I²C 驱动中默认的 I²C 频率都是 100KHZ，因此需要根据实际需求来设置 I²C 频率。

如使用 I²C 连接 sensor 和主芯片，并且 sensor 的 I²C 接口支持 400KHZ 频率时，我们就需要修改默认频率。详见图 2-4，修改点为 i2c-hisilicon.c 中的 hi_i2c_platform_data 结构体：

图2-4 I²C 频率修改

```
struct hi_platform_i2c hi_i2c_platform_data[] = {
    [0] =
        {
            .clk_limit    = 100000,
            .i2c_class    = I2C_CLASS_DDC,
            .clk_rate     = CONFIG_HI_I2C0_APB,
        },
    [1] =
        {
            .clk_limit    = 100000,
            .i2c_class    = I2C_CLASS_DDC,
            .clk_rate     = CONFIG_HI_I2C1_APB,
        },
    [2] =
        {
            .clk_limit    = 100000,
            .i2c_class    = I2C_CLASS_DDC,
            .clk_rate     = CONFIG_HI_I2C2_APB,
        },
#if I2C_NUM >= 4
    [3] =
        {
            .clk_limit    = 100000,
            .i2c_class    = I2C_CLASS_DDC,
            .clk_rate     = CONFIG_HI_I2C2_APB,
        },
#endif
};
```

说明

如果是 sensor 初始化已经放到 uboot 中的，需要修改 uboot 中的 i2c 频率。

3 快速启动 Sample 说明

3.1 概述

为了帮助用户理解 Huawei LiteOS 里快速启动所适用的各种优化措施，在 SDK 发布包里，提供了一个快速启动的 Sample 工程。

以 Hi3516A 为例，使用到的优化措施有：

1. 使用 mini uboot。
2. 将 sensor 提前到 uboot 里初始化，且 I^2C 时钟提高到 400kHz。
3. SPI Flash 驱动优化。
4. 使用分散加载特性。
5. 关闭 Huawei LiteOS 的打印。

其他的优化措施除了硬件解压外，都是需要用户自己在 APP 里进行的。所以此处不提。

3.2 支持的硬件

Sample 工程可以在 Hi3516A、Hi3518EV20X 的 DEMO 板上运行。之后会持续添加对新的硬件单板的支持。

3.3 工程目录说明

将快速启动 Sample 工程放在 SDK 发布包目录下的 sample 目录里，所在子目录名为 quickstart。该目录下的目录及文件说明如表 3-1 所示。

表3-1 Sample 工程目录结构说明

目录或文件名	说明
deps	依赖的头文件和库文件，主要是针对快速启动修改后的 sdk 库和 Huawei LiteOS 库文件。Huawei LiteOS 的库主要替换去掉打印的。

目录或文件名	说明
out	编译中间文件、输出文件及最终镜像。烧录镜像在 burn 目录下，名字为 sample.bin。
script	相关脚本。
src	源代码目录。
tools	相关工具。
uboot	mini uboot 源代码。
cfg.mak	工程的配置文件。
burn-guild.txt	烧录说明。
Makefile	工程的主 Makefile。

3.4 各组件改动点

快速启动需要修改的各个组件及其主要内容如表 3-2 所示。

表3-2 Sample 工程各个组件的改动点

组件	改动点
uboot	• 裁减成 mini uboot。 • 将 sensor 提前到 uboot 里初始化。
OS	关闭打印功能。
SDK	因为 sensor 初始化提前到 uboot 里执行了，所以 SDK 里的对应代码需要修改。修改后的内容，也是编译成库文件放在 deps/lib 目录下。
APP	• sdk_init.c 里，SDK 的初始化过程，有些函数需要注释掉，以避免对 sensor 的重新初始化。具体改动点，请对比 mpp/init/sdk_init.c 和 mpp/sample/quickstart/src/media/sdk_init.c 这 2 个文件。 • 使用分散加载特性。

下面依次是各个组件的详细改动说明。

3.4.1 uboot 改动

mini uboot 的裁剪内容较多，不便在此详细说明。详细内容请参考对应的文档。

3.4.2 OS 改动

用户如果可以拿到 Huawei LiteOS 的源代码，则可以修改打印函数的实现，来直接屏蔽掉系统中的打印，以加快系统启动的速度。改动内容如表 3-3 所示。

表3-3 OS 改动示例

文件名	修改示例

liteos/platform/bsp/common/platform_printf.c	将 dprintf 函数改成如下所示。
	`__attribute__ ((noinline)) void dprintf(const char *`
	`{`
	`  if(debug_on)`
	`  {`
	`  int uwlen;`
	`  char abuf[SIZEBUF];`
	`  memset(abuf,0 , SIZEBUF);`
	`  va_list ap;`
	`  va_start(ap, fmt);  /*lint !e1055 !e534 !e53`
	`  uwlen = __dprintf(fmt, ap, (void*) uart_fputc,`
	`#ifdef LOSCFG_NET_TELNET`
	`  if(telnet_Mask() == 1) {`
	`  write(telnet_client_fd,abuf,uwlen);`
	`  }`
	`#endif`
	`  va_end(ap);`
	`  }`
	`} ? end  attribute  ?`

3.4.3 SDK 改动

SDK 的改动如表 3-4 所示。

表3-4 SDK 改动点

文件名	改动点
drv/interdrv/mipi/mipi.c	找到 mipi_set_combo_dev_attr 函数，在函数的最前面直接返回成功。不执行函数内容。
mpp\component\isp\sensor\omnivision_ov9732\ov9732_sensor_ctl.c	找到 sensor_init 函数，在函数最前面直接返回。 omnivision_ov9732\ov9732_sensor_ctl.c 这个跟具体板子使用的 sensor 有关。当前以 ov9732 举例。

3.4.4 APP 改动

app 代码在 quickstart/src 目录下。

表3-5 APP 改动说明

文件名或者目录	说明

sample_quickstart.c	本文件是快速启动 sample 的入口文件。
	主体代码如下图所示。
	`void app_init(void)`
	`{`
	`misc_driver_init();`
	`SDK_init();`
	`print_time(__func__, __LINE__, "sdk init finish");`
	`media_init();`
	`#ifdef CFG_SCATTER_FLAG`
	`LOS_ScatterLoad(0x100000, image_read, 1);//"0x100000" is the addre`
	`#endif`
	`#ifndef CFG_FAST_IMAGE`
	`//User can place business code here.`
	`dprintf("------------------start finish------------------ \n");`
	`#ifdef CFG_START_SNAP`
	`do_snap();`
	`#endif`
	`proc_fs_init();`
	`osShellInit();`
	`shell_cmd_register();`
	`#endif`
	`while (1)`
	print_time 是用来打印启动过程中时间点的函数，即使在关闭 OS 的打印后，这个函数的打印仍然有效。

3.5 编译及烧录步骤

编译及烧录步骤具体如下：

步骤 1. 进入 quickstart 目录，输入 make liteos_image。这个编译命令的作用是生成可以分散加载的镜像文件。关于这个命令的执行过程，用户可以参考《Huawei LiteOS 分散加载应用说明》。

步骤 2. 等编译过程完成。

步骤 3. 具体烧录过程，请参考 burn-guide.txt 文件。

----结束

3.6 Sample 注意事项

表 3-6 列出了 sample 工程可能碰到的问题及解决方法。

表3-6 Sample 问题分析及解决

问题及注意点	分析及解决
在 quickstart 目录下执行了 make liteos_image 后，再去编译其他的 sample 工程，会出现编译不过的问题。	分析： 因为 quickstart 的编译过程使用了分散加载特性，这一特性会在编译过程中修改 liteos/tools/scripts/ld 目录下的 wow.ld 和 scatter.ld 这 2 个链接脚本。而 SDK 的其他 sample 是没有使用这一特性的。所以此时需要将 liteos 的环境进行还原。 解决： 在 quickstart 目录下，输入 make liteos_image_clean 命令。这个命令的作用是将 liteos/tools/scripts/ld 目录下的链接脚本还原。此时再编译 SDK 其他的 sample 即可。