



Linux D1 开发指南

版本号: 1.6

发布日期: 2024.06.11

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.05.24	AWA1639	初始版本
1.1	2022.7.12	AWA1836	添加支持 linux-5.10
1.2	2022.10.31	AWA1442	添加 DI200 相关描述
1.3	2023.03.06	AWA1729	添加 DI300 支持 linux-5.15
1.4	2023.08.01	AWA1729	添加 T527 支持 DI300 的信息以及 PM 框架的依赖
1.5	2023.11.15	AWA1729	1. DI300 和 DI200 分开为 2 个章节来描述 2. 文档勘误 3. 增加 DI110 章节
1.6	2024.06.11	AWA1729	FAQ 中更新动态日志打印功能描述

目 录

1 概述	1
1.1 编写目的	1
1.2 适用范围	1
1.3 相关人员	1
1.4 相关术语介绍	2
2 DI300 模块	3
2.1 模块介绍	3
2.1.1 模块功能介绍	3
2.1.2 模块配置介绍	4
2.1.2.1 设备树	4
2.1.2.2 menuconfig	4
2.1.3 源码结构介绍	6
2.2 模块接口说明	7
2.2.1 驱动关键结构体说明	7
2.2.1.1 struct di_timeout_ns	7
2.2.1.2 struct di_dit_mode	8
2.2.1.3 struct di_tnr_mode	8
2.2.1.4 di_fmd_enable	8
2.2.1.5 struct di_size	8
2.2.1.6 struct di_buf	9
2.2.1.7 struct di_fb	9
2.2.1.8 struct di_process_fb_arg	9
2.2.2 模块使用接口说明	10
2.2.2.1 DI_IOC_RESET	10
2.2.2.2 DI_IOC_CHECK_PARA	10
2.2.2.3 DI_IOC_SET_TIMEOUT	11
2.2.2.4 DI_IOC_SET_VIDEO_SIZE	11
2.2.2.5 DI_IOC_SET_DIT_MODE	11
2.2.2.6 DI_IOC_SET_TNR_MODE	11
2.2.2.7 DI_IOC_SET_FMD_ENABLE	11
2.2.2.8 DI_IOC_PROCESS_FB	12
2.2.3 驱动调用流程说明	12
2.2.4 模式介绍及使用说明	13
2.2.4.1 60Hz 模式	13
2.2.4.2 30Hz 模式	15
2.2.4.3 BOB 模式	17
2.2.4.4 TNROnly 模式	18

3	DI200 模块	20
3.1	模块介绍	20
3.1.1	模块功能介绍	20
3.1.2	模块配置介绍	20
3.1.2.1	设备树	20
3.1.2.2	menuconfig	21
3.1.3	源码结构介绍	21
3.2	模块接口说明	22
3.2.1	驱动关键结构体说明	22
3.2.1.1	di_struct	22
3.2.1.2	di_mem_t	23
3.2.1.3	__di_para_t	23
3.2.1.4	__di_fb_t	23
3.2.1.5	__di_para_t2	24
3.2.1.6	__di_fb_t2	24
3.2.1.7	__di_rectsz_t	24
3.2.1.8	__di_mode_t	25
3.2.2	模块使用接口说明	25
3.2.2.1	DI_IOCSTART	25
3.2.2.2	DI_IOCSTART2	25
3.2.2.3	DI_IOCSETMODE	25
3.2.3	驱动调用流程说明	26
3.2.4	模式介绍及使用说明	26
4	DI110 模块	27
4.1	模块功能	27
4.1.1	模块功能介绍	27
4.1.2	模块规格介绍	27
4.1.3	模块配置介绍	27
4.1.3.1	设备树	27
4.1.3.2	menuconfig	28
4.1.4	源码结构介绍	29
4.2	模块接口说明	30
4.2.1	驱动关键结构体说明	30
4.2.1.1	struct di_version	30
4.2.1.2	struct di_mem_arg	30
4.2.1.3	struct di_timeout_ns	30
4.2.1.4	struct di_tnr_mode	30
4.2.1.5	struct di_size	31
4.2.1.6	struct di_addr	31
4.2.1.7	struct di_offset	31
4.2.1.8	union di_buf	31
4.2.1.9	struct di_fb	32

4.2.1.10	struct di_process_fb_arg	32
4.2.2	模块使用接口说明	33
4.2.2.1	DI_IOC_GET_VERSION	33
4.2.2.2	DI_IOC_SET_TIMEOUT	33
4.2.2.3	DI_IOC_PROCESS_FB	33
4.2.2.4	DI_IOC_MEM_REQUEST	34
4.2.2.5	DI_IOC_MEM_RELEASE	34
4.2.3	应用层调用流程说明	34
4.2.3.1	软件架构	34
4.2.3.2	帧处理流程	36
4.2.4	DI 模式介绍及使用说明	36
4.2.4.1	BOB 单帧模式	37
4.2.4.2	BOB 双帧模式	37
4.2.4.3	MD 单帧模式	38
4.2.4.4	MD 双帧模式	39
5	FAQ	41
5.1	常用调试方法	41
5.1.1	DI110/DI300 动态调整驱动日志打印等级	41
5.1.1.1	对模块的属性节点进行参数设置	41
5.1.1.2	使用 AW 日志基础组件	41
5.1.2	DI200 动态调整驱动日志打印等级	42

插图

图 2-1	DI menuconfig	5
图 2-2	DI menuconfig-5.10	6
图 2-3	DI300_Help 页面	6
图 2-4	DI 驱动结构	7
图 2-5	驱动调用流程	12
图 2-6	60 Hz 数据流模型	13
图 2-7	60 Hz 起播流程	14
图 2-8	60 Hz 时序数据流	14
图 2-9	60 Hz 结束流程	15
图 2-10	30Hz 数据流模型	16
图 2-11	30Hz 起播流程	16
图 2-12	30Hz 时序数据流	16
图 2-13	30Hz 结束流程	17
图 2-14	dit BOB 数据流模型	18
图 2-15	dit tnr 数据流模型	18
图 2-16	dit tnr 起播流程	19
图 2-17	dit tnr 时序数据流	19
图 3-1	DI menuconfig-5.10	21
图 3-2	DI 驱动结构	22
图 4-1	DI menuconfig	28
图 4-2	DI menuconfig_5.10 及以上版本	28
图 4-3	DI 驱动结构	29
图 4-4	软件总体架构	34
图 4-5	驱动调用流程图	35
图 4-6	帧处理流程图	36
图 4-7	BOB 单帧模式数据流模型	37
图 4-8	BOB 双帧模式数据流模型	38
图 4-9	MD 单帧模式数据流模型	39
图 4-10	MD 双帧模式数据流模型	40
图 5-1	确认 DI 的动态日志功能是否使能	42

1 概述

1.1 编写目的

介绍 Linux 内核中去隔行模块（De-interlace，简称为 DI）驱动接口及使用方法，为本模块的使用者提供参考。

1.2 适用范围

表 1-1: 适用产品列表

产品名称	硬件版本	内核版本	驱动文件路径
T507	DI300	Linux-4.9/ Linux-5.4	drivers/char/sunxi-di/drv_div3x
T507	DI300	Linux-5.10	bsp/drivers/di/drv_div3x
H616/ H618	DI300	Linux-4.9/ Linux-5.4	drivers/char/sunxi-di/drv_div3x
A523/ T527/ T527V	DI300	Linux-5.15	bsp/drivers/di/drv_div3x
T3/T3-C/ T3PRO	DI200	Linux-5.10	bsp/drivers/di/drv_div2x
A40I-C/ A40I-H	DI200	Linux-5.10	bsp/drivers/di/drv_div2x && bsp/drivers/di/ drv_div2x/lowlevel_v1x/
T113	DI110	Linux-5.4/ Linux-5.15	drivers/char/sunxi-di/drv_div1xx && bsp/ drivers/di/drv_div1xx

1.3 相关人员

- DI 驱动及应用层开发工程师
- 技术支持工程师

1.4 相关术语介绍

表 1-2: 术语介绍

术语	解释说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台。
DI	De-Interlace 的缩写，即反交错或者去隔行。把隔行扫描的图像恢复为逐行扫描的图像。
DIT	实现 De-Interlace 的硬件模块。
TNR	实现 3D 降噪的硬件模块。
FMD	实现电影模式检测的硬件模块。
MD	实现运动检测的硬件模块。
democrop	指实现画面对比的功能，demo 窗口内作 DIT/TNR 处理，demo 窗口外部不做 DIT/TNR 处理。
top/bottom field	行数号从 0 开始。行数号为偶数的场为偶数场，称为 top field。奇数场称为 bottom field。
first field	时间戳小的为 first field(首场)，时间戳大的为次场。
top field first	指偶数场是首场。
base field	基场，指在输出帧内容是基于输入帧的哪一场，可以是 top field，也可以是 bottom field。输出帧与其基场的内容最接近。
I 源视频	指隔行扫描的视频
P 源视频	指逐行扫描的视频

2 DI300 模块

2.1 模块介绍

2.1.1 模块功能介绍

DI300 去隔行模块，除了可以把隔行扫描的图像恢复为逐行扫描的图像之外，还提供了丰富的功能为视频播放提供更优秀的效果表现。

1. 支持视频分辨率为 32x32 到 2048x1280 pixel。
2. 支持去隔行（DIT）功能。
3. 支持运动检测（MD）功能实现更好的 DIT 效果。
4. 支持 3D 降噪（TNR）功能。
5. 支持电影模式检测（FMD）功能。
6. 支持 democrop 画面对比功能。
7. DI300 支持 NV12/NV21/YV12/YUV422/YUV422 UV-combined 格式的输入。
8. 支持多种工作模式，包括 BOB、TNROnly、30 Hz、60 Hz 四种工作模式。
9. BOB 模式下支持 NV12/NV21/YV12/YUV422/YUV422 UV-combined 格式的输出。
10. 60 Hz/30 Hz/TNROnly 模式下仅支持 YV12/YUV422 格式的输出。

2.1.2 模块配置介绍

2.1.2.1 设备树

```
di:deinterlace@5400000 {
    #address-cells = <1>;
    #size-cells = <0>;
    compatible = "allwinner,sunxi-deinterlace";
    reg = <0x0 0x05400000 0x0 0x040000>;
    interrupts = <GIC_SPI 88 IRQ_TYPE_LEVEL_HIGH>;
    iommu = <&mmu_aw 6 1>;
    power-domains = <&pd1 A523_PCK_V00>;
    status = "okay";

    clocks = <&ccu CLK_DI>, <&ccu CLK_BUS_DI>, <&ccu CLK_PLL_VIDEO0_4X>;
    clock-names = "clk_di", "clk_bus_di", "clk_di_parent";
    clock-frequency = <300000000>;

    resets = <&ccu RST_BUS_DI>;
    reset-names = "rst_bus_di";
};
```

如上，在 DI 设备树中主要是配置时钟中断寄存器 iommu 等硬件资源。

2.1.2.2 menuconfig

在命令行中进入 longan 根目录，执行./build.sh menuconfig 进入配置主界面，如果芯片使用的 DI 硬件是 DI300，选上” Support DI V300 “即可。

2.1.2.2.1 Linux-5.10 以下版本

```
Device Drivers --->
<*> Allwinnertech DE-Interlace Driver --->
  [ ] Support DI V110/1200
  [ ] Support DI V200
  [*] Support DI V300
```

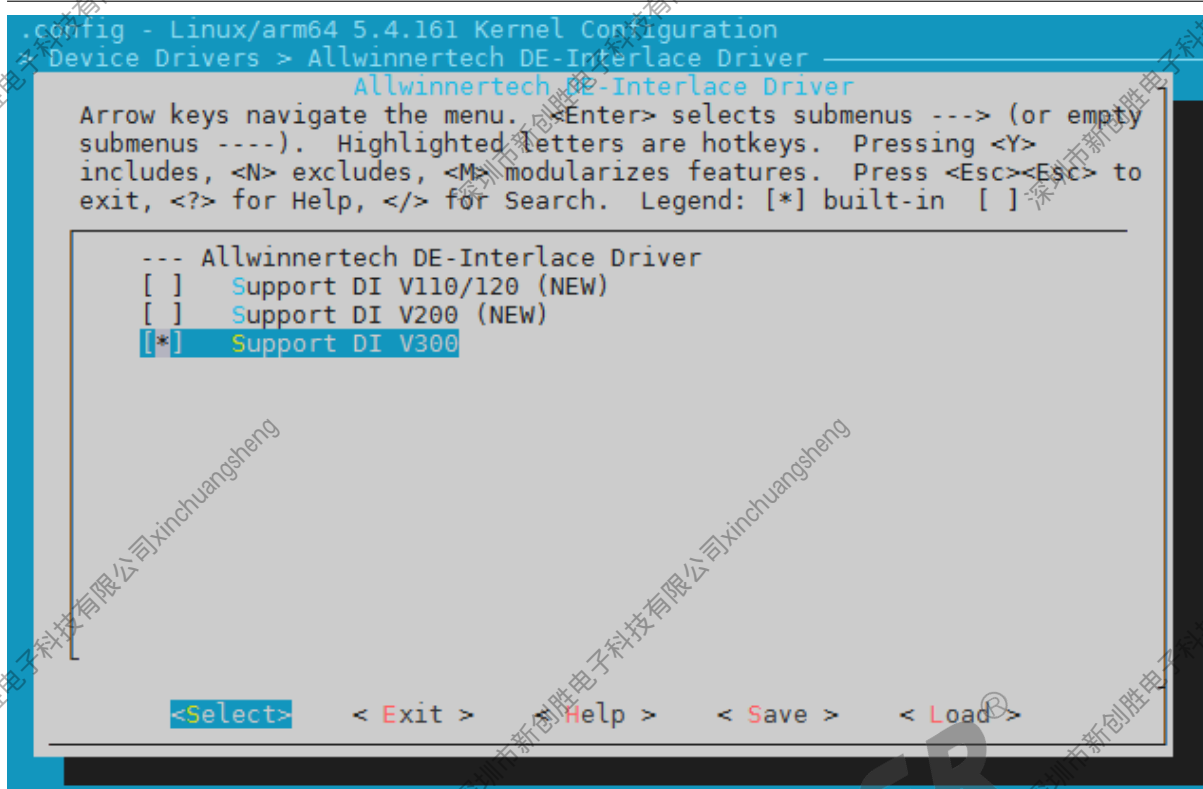


图 2-1: DI menuconfig

2.1.2.2.2 Linux5.10 及以上版本

```
Allwinner BSP --->
Device Drivers --->
DI_Drivers --->
[*] DI Support for Allwinner SoCs
[ ] Support DI V110/120
[ ] Support DI V200
[*] Support DI V300
[*] Support Single File
[ ] Open amp server
```

📖 说明

1. **Support Single File** 指的是支持输入的图像顶场和底场分别由两个 buffer 存储，一般正常开启即可。（较老版本客户的 demo 可能不支持 single file，由于该功能会修改到 ioctl 的传入参数，如果遇到 ioctl 的传入参数不符问题，可以考虑尝试关掉这个功能）。
2. **Open amp server** 指在驱动中打开 amp 服务，用于多核异构场景下使用。

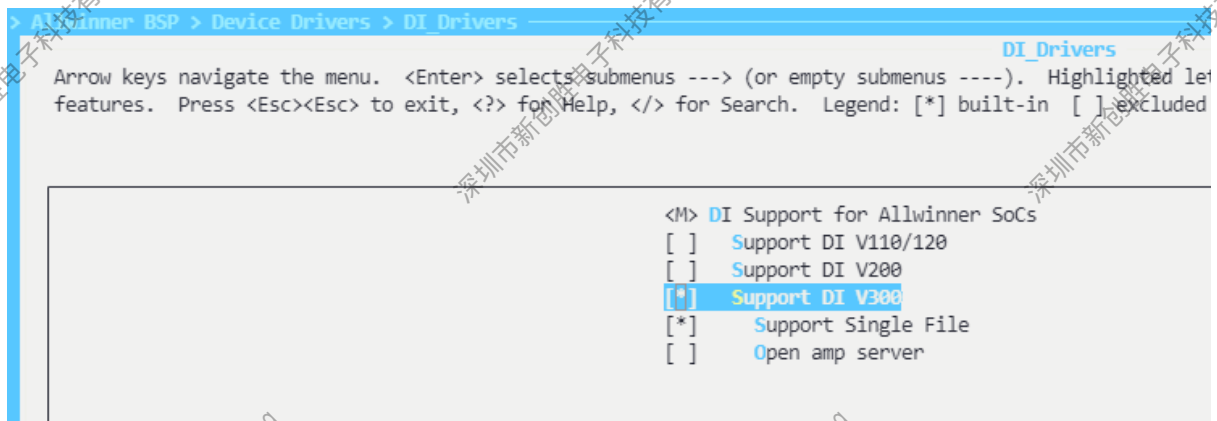


图 2-2: DI menuconfig-5.10

注：从 5.10 版本开始，DI300 驱动依赖 PM（Power Manager）驱动。所以在选中 SUNXI_DI_V3X 之前，需要让 PM 和 PM_SLEEP 都置为 Y 才能满足前置依赖项

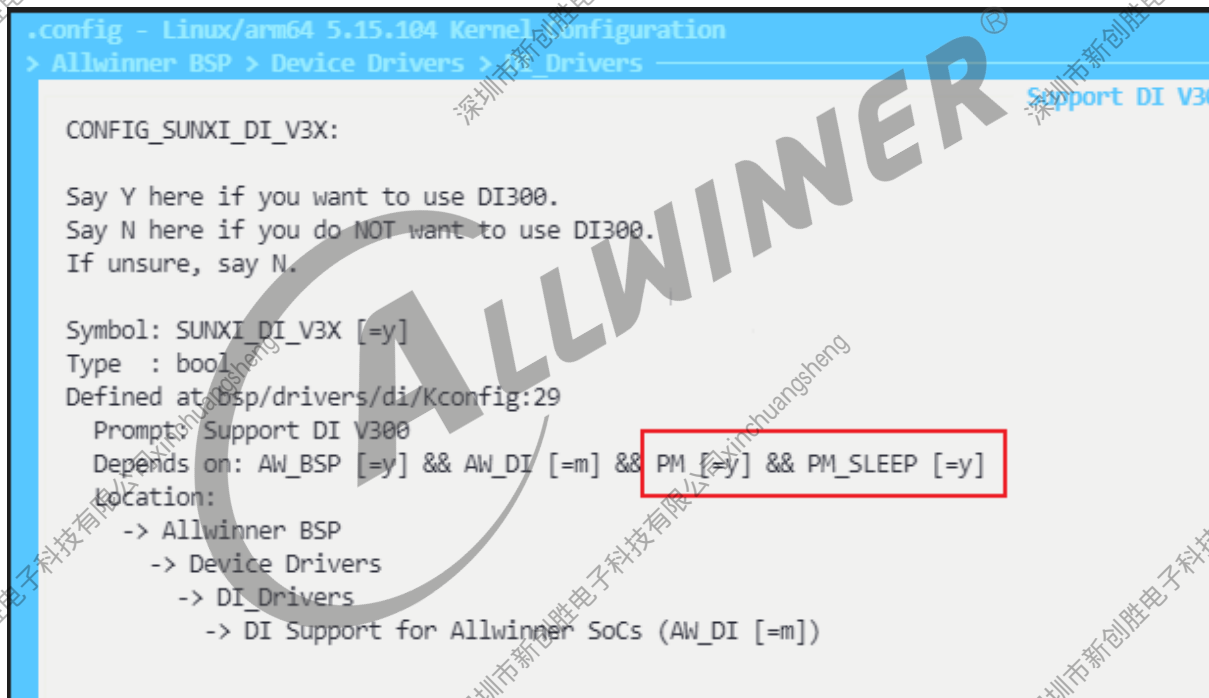


图 2-3: DI300_Help 页面

2.1.3 源码结构介绍

- |— di_client.c
- |— di_client.h
- |— di_debug.h
- |— di_dev.h
- |— di_driver.c
- |— di_driver.h
- |— di_fops.c

```
├── di_fops.h
├── di_utils.c
├── di_utils.h
├── lowlevel_v3x
│   ├── di300_alg.c
│   ├── di300_alg.h
│   ├── di300.c
│   ├── di300.h
│   └── di300_reg.h
├── Makefile
└── sunxi_di.h
```

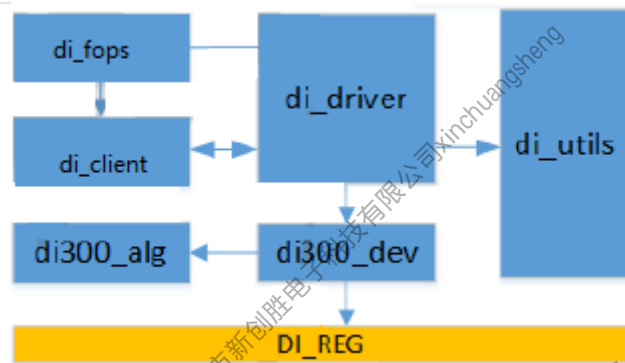


图 2-4: DI 驱动结构

- di_driver: 驱动的核心，实现驱动加载、资源管理、多用户复用调度、中断处理等。
- di_fops: 对接用户空间的文件 io 操作。
- di_client: 负责 client 的资源管理，如 di 模式参数、fb 参数等。
- di_utils: 提供 dma_buf 封装接口、fourcc_format 管理接口等。
- di300_dev: 配置寄存器操作。
- di300_alg: 实现了电影模式检测、场检测等软件算法。

2.2 模块接口说明

2.2.1 驱动关键结构体说明

2.2.1.1 struct di_timeout_ns

定义等待超时相关的时间，时间单位为 ns。

成员	数据类型	I/O	描述
wait4start	u64	输入	等待另一用户完成处理后获取 di 硬件资源的超时时间。默认值为 3s
wait4finish	u64	输入	等待 di 硬件完成处理的超时时间。默认值为 3s

2.2.1.2 struct di_dit_mode

描述 dit 模式的相关定义。

成员	数据类型	I/O	描述
intp_mode	u32	输入	di 处理的插值模式，值为 DI_DIT_INTP_MODE_xxx
out_frame	u32	输入	dit 输出帧数，值为 DI_DIT_OUT_xFRAME

2.2.1.3 struct di_tnr_mode

描述 tnr 模式的相关定义。

成员	数据类型	I/O	描述
mode	u32	输入	去噪模式，值为 DI_TNR_MODE_xxx
level	u32	输入	自适应去噪模式下的强度等级

2.2.1.4 di_fmd_enable

描述 fmd 功能使能的定义。

成员	数据类型	I/O	描述
en	u32	输入	置 1 表示打开 fmd 功能

2.2.1.5 struct di_size

一般用于描述图像的分辨率，单位 pixel。

成员	数据类型	I/O	描述
width	u32	输入	宽
height	u32	输入	高

2.2.1.6 struct di_buf

描述 buf 相关信息。

成员	数据类型	I/O	描述
y_addr	u64	输入	y 分量的首物理地址, 或地址偏移量
cb_addr	u64	输入	cb 分量的首物理地址, 或地址偏移量, 或 0
cr_addr	u64	输入	cr 分量的首物理地址, 或地址偏移量, 或 0
ystride	u32	输入	y 分量的线宽, 单位 byte
cstride	u32	输入	cb/cr 分量的线宽, 单位 byte

说明

1. 当使用物理地址方式时, ycbcr addr 为首物理地址; 当使用 dmabuf 方式时, ycbcr addr 为地址偏移量。
2. 当像素格式为 uvcombined 时, 如果 cr 分量为先, 则 cb_addr 为 0, 反之 cr_addr 为 0。

2.2.1.7 struct di_fb

描述一帧 Framebuffer 的相关信息。

成员	数据类型	I/O	描述
format	u32	输入	视频帧的像素格式, 详见 fourcc 定义
dma_buf_fd	s32	输入	dmabuf 文件描述符, 小于 0 为无效值
buf	struct	输入	di_buf
size	struct	输入	di_size

说明

当 dma_buf_fd 为无效值时, 表示使用物理地址方式, 否则使用 dmabuf 方式。

2.2.1.8 struct di_process_fb_arg

描述一次 di process 所需的信息。

成员	数据类型	I/O	描述
is_interlace	u8	输入	1 为 I 源视频, 0 为 P 源视频
is_pulldown	u8	输入	暂无使用
top_field_first	u8	输入	1 表示 top_field 的时间戳较小, 0 表示 bottom_field 的时间戳较小
base_field	u8	输入	指定 BOB 模式的基场。0 表示 top_field, 1 表示 bottom_field

成员	数据类型	I/O	描述
in_fb0	struct di_fb	输入	时域上的第 1 帧（或者是第 1 帧的首场）输入 fb
in_fb1	struct di_fb	输入	时域上的第 2 帧（或者是第 2 帧的首场）输入 fb
in_fb2	struct di_fb	输入	时域上的第 3 帧（或者是第 3 帧的首场）输入 fb
in_fb0_nf	struct di_fb	输入	时域上的第 1 帧的次场输入 fb
in_fb1_nf	struct di_fb	输入	时域上的第 2 帧的次场输入 fb
in_fb2_nf	struct di_fb	输入	时域上的第 3 帧的次场输入 fb
out_dit_fb0	struct di_fb	输入	时域上的第 1 帧输出 fb
out_dit_fb1	struct di_fb	输入	时域上的第 2 帧输出 fb
out_tnr_fb0	struct di_fb	输入	tnr 输出 fb

说明

1. 只有在驱动开启“Support Single File”选项时，struct di_process_fb_arg 才会有 in_fb0_nf、in_fb1_nf、in_fb2_nf 这 3 个成员。
2. 输入的帧数据是单场时：in_fb0、in_fb1、in_fb2 表示输入帧数据的首场，而 in_fb0_nf、in_fb1_nf、in_fb2_nf 表示为输入帧数据的次场。
3. 如果输入的帧数据是完整的一帧时，不需要给 in_fb0_nf、in_fb1_nf、in_fb2_nf 赋值，仅需要给 in_fb0、in_fb1、in_fb2 赋值即可。

2.2.2 模块使用接口说明

DI300 驱动使用标准的文件 IO 接口：open、close、ioctl。下面描述 ioctl 的命令接口。

2.2.2.1 DI_IOCTL_RESET

synopsis ioctl(fd, DI_IOCTL_RESET, long)
arguments null
returns
Zero, if successful.
Negative, if there is an error.
description This api reset all setting parameters in software.

2.2.2.2 DI_IOCTL_CHECK_PARAM

synopsis ioctl(fd, DI_IOCTL_CHECK_PARAM, long)
arguments null
returns
Zero, if successful.
Negative, if there is an error.
description This api check the setting parameters.

2.2.2.3 DI_IOC_SET_TIMEOUT

synopsis `ioctl(fd, DI_IOC_SET_TIMEOUT, struct di_timeout_ns *)`
arguments `struct di_timeout_ns *`
returns
Zero, if successful.
Negative, if there is an error.
description This api set timeout values.

2.2.2.4 DI_IOC_SET_VIDEO_SIZE

synopsis `ioctl(fd, DI_IOC_SET_VIDEO_SIZE, struct di_size *)`
arguments `struct di_size *`
returns
Zero, if successful.
Negative, if there is an error.
description This api set resolution size of video.

2.2.2.5 DI_IOC_SET_DIT_MODE

synopsis `ioctl(fd, DI_IOC_SET_DIT_MODE, struct di_dit_mode *)`
arguments `struct di_dit_mode *`
returns
Zero, if successful.
Negative, if there is an error.
description This api set dit mode.

2.2.2.6 DI_IOC_SET_TNR_MODE

synopsis `ioctl(fd, DI_IOC_SET_TNR_MODE, struct di_tnr_mode *)`
arguments `struct di_tnr_mode *`
returns
Zero, if successful.
Negative, if there is an error.
description This api set tnr mode.

2.2.2.7 DI_IOC_SET_FMD_ENABLE

synopsis `ioctl(fd, DI_IOC_SET_FMD_ENABLE, struct di_fmd_enable *)`
arguments `struct di_fmd_enable *`
returns
Zero, if successful.
Negative, if there is an error.
description This api enable/disable fmd function.

2.2.2.8 DI_IOC_PROCESS_FB

synopsis `ioctl(fd, DI_IOC_PROCESS_FB, struct di_process_fb_arg *)`

arguments `struct di_process_fb_arg *`

returns

Zero, if successful.

Negative, if there is an error.

description This api do process fb of DI.

2.2.3 驱动调用流程说明

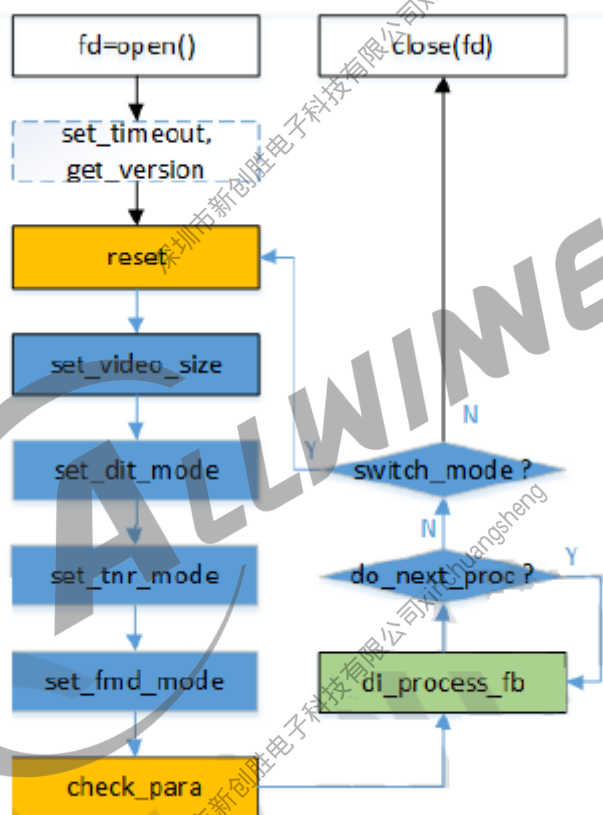


图 2-5: 驱动调用流程

1. open 打开 DI 模块，使用结束时使用 close 关闭 DI 模块。
2. set_timeout（设置超时等待值）、get_version（获取软硬件版本号），可选项。
3. 调用 reset 接口，进行软件的模式参数的 reset。必选项。
4. set_video_size 设置视频分辨率，必选项。

5. `set_xxx_mode` 设置 DI 各模块的模式参数。必须至少设置 1 个模块的模式参数。后续章节详细介绍具体模式下的模式参数值。
6. 设置视频分辨率和模式参数后，调用 `check_para` 接口进行参数检测，必选项。
7. 循环调用 `di_process_fb` 进行 framebuffer 处理。后续章节详细介绍具体模式下的 Framebuffer 的参数要求。
8. 如果需要模式切换（比如从 BOB 模式切换到 60 Hz 模式），有两种方式。第一种是 `close` 后再 `open`，重新进行设置流程。第二种方式是调用 `reset` 接口，再重新进行设置流程。

2.2.4 模式介绍及使用说明

2.2.4.1 60Hz 模式

- 60 Hz 模式为 DIT/TNR/FMD 功能全打开的模式，具有去隔行功能、3D 降噪和电影模式、运动检测，效果最好。
- 60 Hz 模式下的每一帧的两场都会处理（非同时），具有 2 倍于 I 源的帧率。

2.2.4.1.1 参数设置

参数	参数值
<code>dit: intp_mode</code>	<code>DI_DIT_INTP_MODE_MOTION</code>
<code>dit: out_frame_mode</code>	<code>DI_DIT_OUT_2FRAME</code>
<code>tnr: mode</code>	<code>DI_TNR_MODE_ADAPTIVE</code> or <code>DI_TNR_MODE_FIX</code>
<code>fmd: enable</code>	1

2.2.4.1.2 数据流基本模型

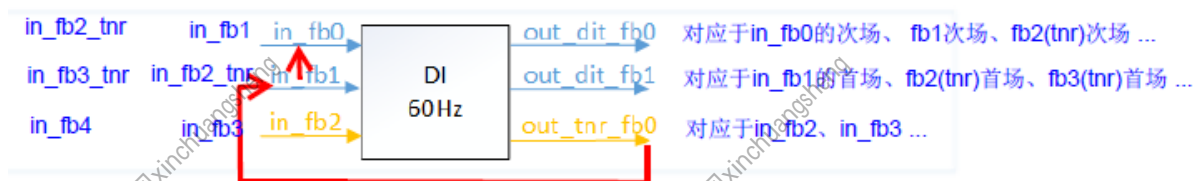


图 2-6: 60 Hz 数据流模型

- out_dit_fb0 对应的基场为 in_fb0 的次场。
- out_dit_fb1 对应的基场为 in_fb1 的首场。

2.2.4.1.3 起播流程

注意，考虑到场次问题，快速起播时：

(1)60Hz模式下，只能选process0→process2;

(2)30Hz模式下，如果选的是out_dit_fb0输出口，则选process0→process2;

(3)30Hz模式下，如果选的是out_dit_fb1输出口，则选process0→process1→process2;

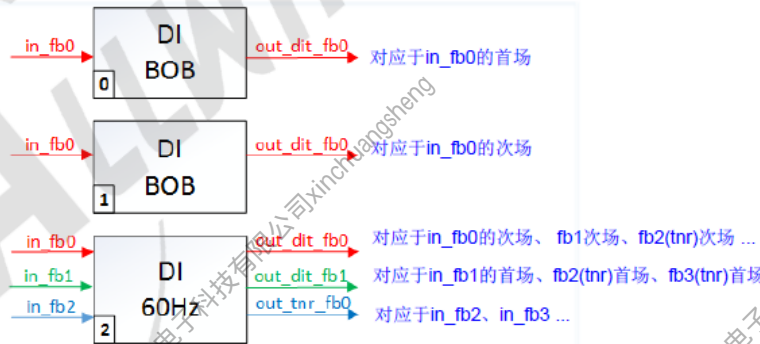


图 2-7: 60 Hz 起播流程

- 注意 process2 的 out_dit_fb0 的基场是 in_fb0(视频首帧) 的次场，所以 process0/1 的 in_fb0 只能是视频首帧，其基场设置需要考虑场序。
- process0/1 是快速起播需求，是可选的。process1 到 process2 之间需要调用 reset 接口。

2.2.4.1.4 时序数据流

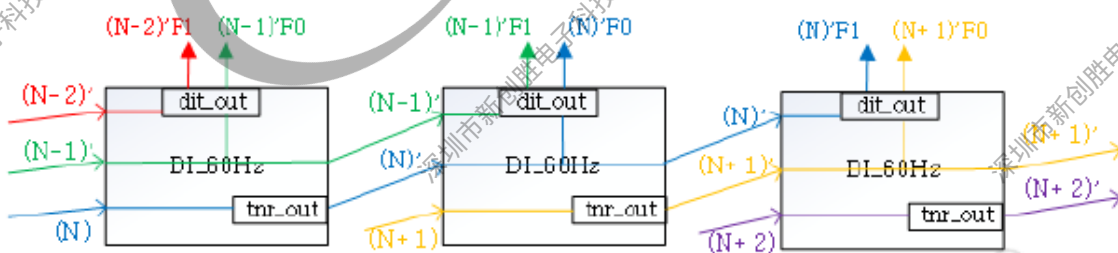


图 8: 60Hz mode 帧流

图 2-8: 60 Hz 时序数据流

2.2.4.1.5 结束流程

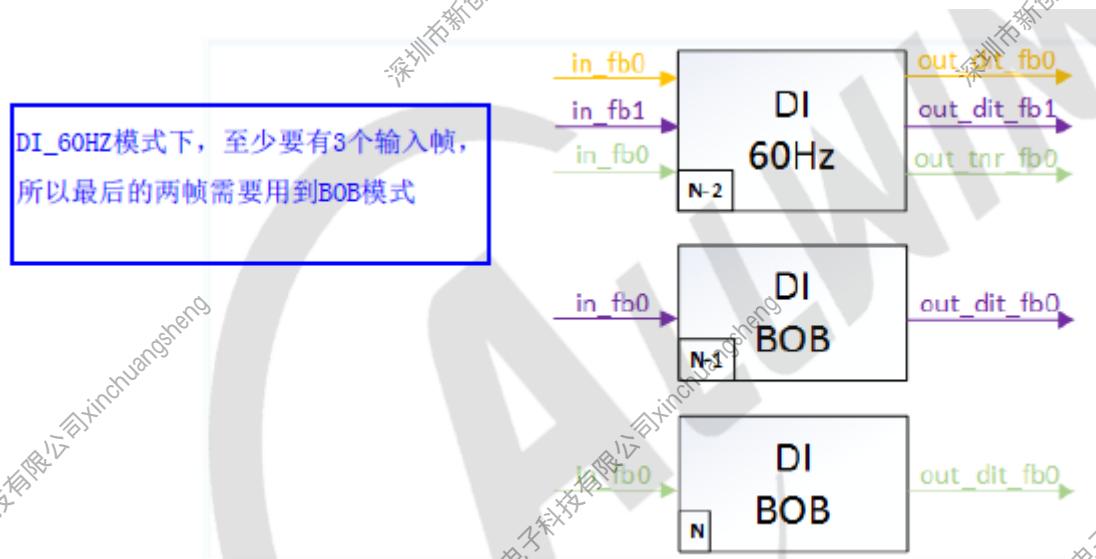


图 2-9: 60 Hz 结束流程

- process(N-2)、process(N-1) 是出最后一帧的需求，是可选的。process(N-2) 到 process(N-1) 之间需要调用 reset 接口。
- 调用者决定最后两帧的基场。如果最后一帧需要出两次，按顺序分别设置基场，调用两次 BOB process fb。

2.2.4.2 30Hz 模式

- 30 Hz 模式为 DIT/TNR/FMD 功能全打开，但 DIT 只处理并输出 1 场。
- 30 Hz 模式具有去隔行功能，同时做 3D 降噪和电影模式、运动检测，效果与 60 Hz 模式一致。
- 30 Hz 模式具有与 I 源相同的帧率。相对于 60 Hz 模式，有利于降低带宽。

2.2.4.2.1 参数设置

参数	参数值
dit: intp_mode	DI_DIT_INTP_MODE_MOTION
dit: out_frame_mode	DI_DIT_OUT_1FRAME
tnr: mode	DI_TNR_MODE_ADAPTIVE or DI_TNR_MODE_FIX
fmd: enable	1

2.2.4.2.2 数据流基本模型

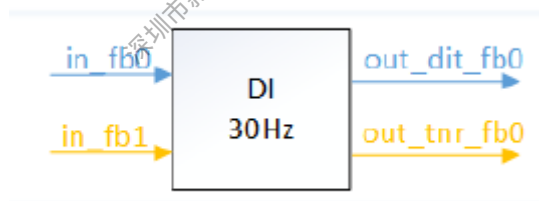


图 2-10: 30Hz 数据流模型

- out_dit_fb0 对应的基场为 in_fb0 的次场

2.2.4.2.3 起播流程



图 2-11: 30Hz 起播流程

- 由于 process1 的 out_dit_fb0 的基场是 in_fb0 的次场，所以 process0 需要把 in_fb0 的基场设置需要考虑场序。
- process0 是快速起播需求，是可选的。process0 到 process1 之间需要调用 reset 接口。

2.2.4.2.4 时序数据流

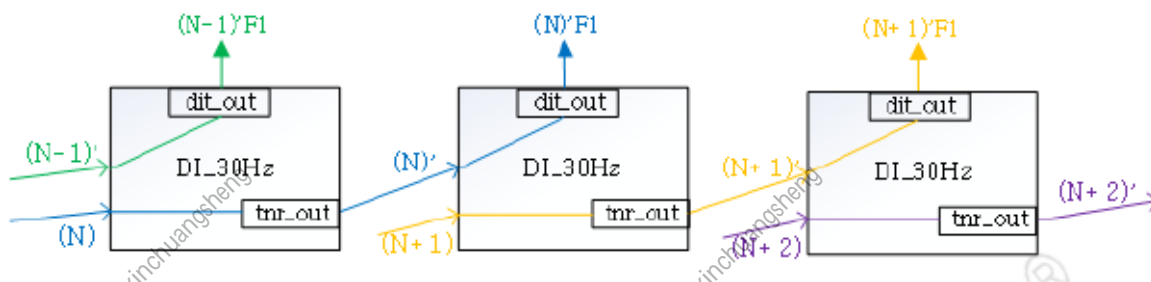


图 2-12: 30Hz 时序数据流

2.2.4.2.5 结束流程

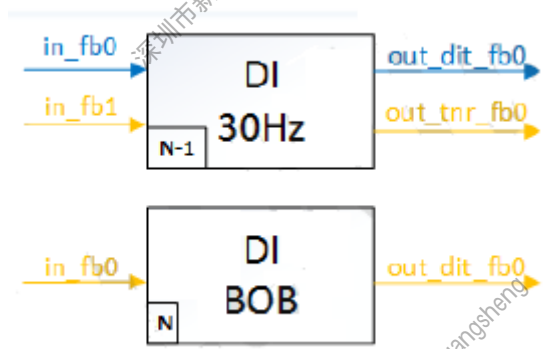


图 2-13: 30Hz 结束流程

- processN 是出最后一帧的需求，是可选的。process(N-1) 到 processN 之间需要调用 reset 接口。
- 调用者决定最后一帧的基场。如果需要出两次，按顺序分别设置基场，调用两次 BOB process fb。

2.2.4.3 BOB 模式

- BOB 是一种单场模式，基于 input fb 的其中一场（奇数场或偶数场）（简称基场），利用场内边缘插值的方法，生成一帧数据并写到 output fb。
- BOB 模式具备去隔行功能，但静止物体的分辨率会下降。BOB 模式的效果比 60 Hz/30 Hz 模式差。
- BOB 模式下的 input fb 的非基场对输出结果无影响。

2.2.4.3.1 参数设置

参数	参数值
dit	intp_mode DI_DIT_INTP_MODE_BOB
dit	out_frame_mode DI_DIT_OUT_1FRAME

2.2.4.3.2 数据流基本模型

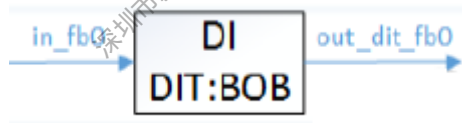


图 2-14: dit BOB 数据流模型

- out_dit_fb0 对应 in_fb0 的基场，基场由 struct di_fb 的成员变量 base_field 决定。
- BOB 模式适用于仅需要去隔行功能，或者需要快速起播的场景：在 30 Hz/60 Hz 模式的正常运行前，使用 BOB 模式处理前 1~2 帧，达到快速出帧的效果。

2.2.4.4 TNROnly 模式

- TNROnly 模式为只打开 3D 降噪功能。不具有去隔行功能。

2.2.4.4.1 参数设置

参数	参数值
tnr: mode	DI_TNR_MODE_ADAPTIVE or DI_TNR_MODE_FIX

2.2.4.4.2 数据流基本模型

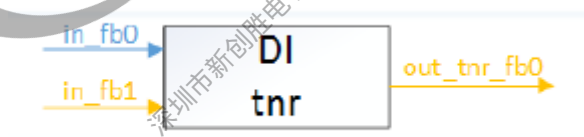


图 2-15: dit tnr 数据流模型

2.2.4.4.3 起播流程

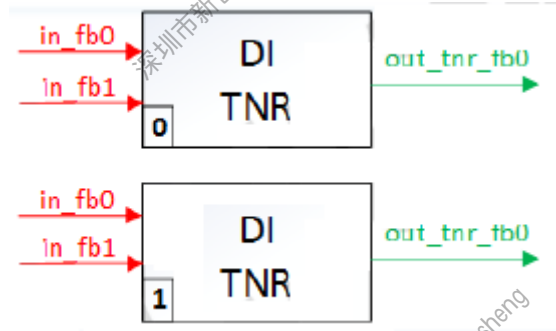


图 2-16: dit tnr 起播流程

- 首帧处理方式：in_fb0 与 in_fb1 为同一帧。

2.2.4.4.4 时序数据流



图 2-17: dit tnr 时序数据流

3 DI200 模块

3.1 模块介绍

3.1.1 模块功能介绍

DI200，即去隔行模块，把隔行扫描的图像恢复为逐行扫描的图像。

1. DI200 lowlevel_v1x 硬件版本只支持 NV12/NV21 格式的输入和输出。
2. DI200 lowlevel_v2x 硬件版本支持 NV12/NV21/YV12/YUV422/YUV422 UV-combined 格式的输入输出。
3. 支持的分辨率为 32x32 到 2048x2048 pixel
4. 支持 DIT 功能

3.1.2 模块配置介绍

3.1.2.1 设备树

```
di:deinterlace@1400000{
    #address-cells = <1>;
    #size-cells = <0>;
    compatible = "allwinner,sunxi-deinterlace";
    reg = <0x0 0x01400000 0x0 0x77c>;
    interrupts = <GIC_SPI 93 IRQ_TYPE_LEVEL_HIGH>;
    status = "okay";

    clocks = <&ccu CLK_DEINTERLACE>,
    <&ccu CLK_BUS_DI>,
    <&ccu CLK_PLL_PERIPH0_1X>,
    <&ccu CLK_DI_GATE>;
    clock-names = "clk_di",
        "clk_bus_di",
        "clk_pll_di",
        "dclk_gate_di";
    resets = <&ccu RST_BUS_DI>;
    reset-names = "rst_bus_di";
```

```
assigned-clocks = <&ccu CLK_DEINTERLACE>;  
assigned-clock-parents = <&ccu CLK_PLL_PERIPH0_1X>;  
};
```

如上，在 DI 设备树中主要是配置时钟中断寄存器 iommu 等硬件资源。

3.1.2.2 menuconfig

在命令行中进入 longan 根目录，执行./build.sh menuconfig 进入配置主界面，如果芯片使用的 DI 硬件是 DI200，选上”Support DI V200 “即可。

3.1.2.2.1 Linux5.10 及以上版本

```
Allwinner BSP --->  
Device Drivers --->  
DI_Drivers --->  
[*] DI Support for Allwinner SoCs  
[ ] Support DI V110/120  
[*] Support DI V200  
[ ] Support DI V300
```

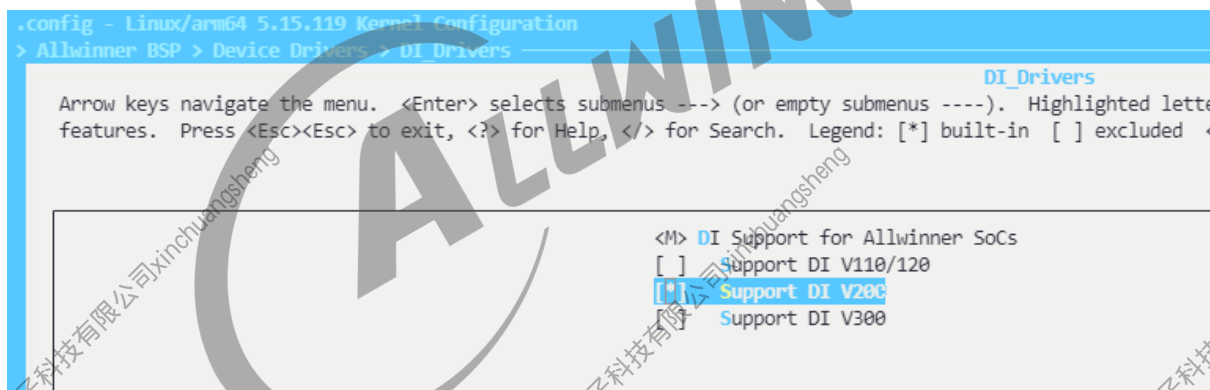


图 3-1: DI menuconfig-5.10

3.1.3 源码结构介绍

```
bsp/drivers/di/drv_div2x/  
├── di.h  
├── lowlevel_v1x  
│   ├── di.c  
│   ├── di_ebios.c  
│   ├── di_ebios_data.h  
│   └── di_ebios.h  
├── lowlevel_v2x  
│   ├── di_type_v2x.h  
│   └── di_v2x.c  
└── Makefile
```

sunxi-di.c
sunxi-di.h

- sunxi_di：驱动的核心，实现驱动加载、资源管理、中断处理等。
- di：封装了操作 lowlevel_v1x 硬件相关的一些接口
- di_v2x.c：对 lowlevel_v2x 硬件的底层寄存器的配置操作
- di_ebios：对 lowlevel_v1x 硬件的底层寄存器的配置操作
- lowlevel_v2x：di200 的其中一个硬件 layout 版本，与 lowlevel_v1x 的差别在于寄存器分布
- lowlevel_v1x：di200 的其中一个硬件 layout 版本

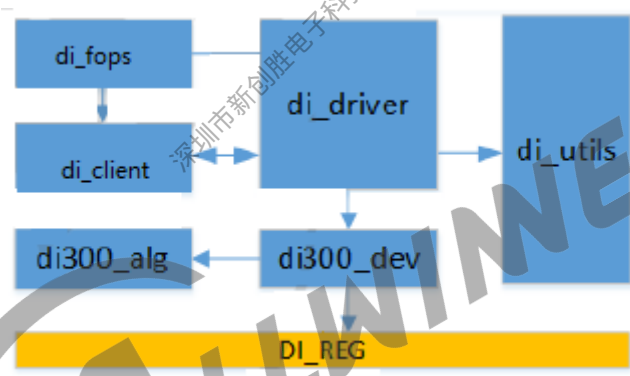


图 3-2: DI 驱动结构

- di_driver：驱动的核心，实现驱动加载、资源管理、多用户复用调度、中断处理等。
- di_fops：对接用户空间的文件 io 操作。
- di_client：负责 client 的资源管理，如 di 模式参数、fb 参数等。
- di_utils：提供 dma_buf 封装接口、fourcc_format 管理接口等。
- di300_dev：配置寄存器操作。
- di300_alg：实现了电影模式检测、场检测等软件算法。

3.2 模块接口说明

3.2.1 驱动关键结构体说明

3.2.1.1 di_struct

描述 di 驱动的相关定义

成员	数据类型	I/O	描述
base_addr	__iomem	输入	di 硬件的寄存器基地址
mem_in_params	struct __di_mem_t	输入	di 输入源的 buffer 信息
mem_out_params	struct __di_mem_t	输入	di 输入源的 buffer 信息
di_complete	atomic_t	输入	di 处理完成的标志
in_flag_phy	void*	输入	使用 motion 方式处理产生的输入场的中间数据
out_flag_phy	void*	输入	使用 motion 方式处理产生的输出场的中间数据
irq_number	u32	输入	di 的软件中断号

3.2.1.2 di_mem_t

描述 di 模块中 buffer 的相关信息

成员	数据类型	I/O	描述
size	unsigned int	输入	buffer 的大小
vaddr	void*	输入	buffer 的虚拟地址
p_addr	unsigned long	输入	buffer 的物理地址

3.2.1.3 __di_para_t

描述 di 场相关的信息

成员	数据类型	I/O	描述
input_fb	struct __di_fb_t	输入	当前场的 fb 信息
pre_fb	struct __di_fb_t	输入	前一场的 fb 信息
output_fb	struct __di_fb_t	输入	输出场的 fb 信息
top_field_first	__u32	输入	首场是否为 top 场标志

3.2.1.4 __di_fb_t

描述场的 fb 相关的信息

成员	数据类型	I/O	描述
addr[2]	void*	输入	存放场的 Y/C 分量数据的物理内存地址
size	__di_rectsz_t	输入	场的像素数据大小
format	__di_pixel_fmt_t	输入	场的格式类型

3.2.1.5 __di_para_t2

描述 di 场相关的信息，与 __di_para_t 作用一致，只是为了兼容不同的 di 硬件 layout，驱动内部会自动对 __di_para_t 和 __di_para_t2 进行转换

成员	数据类型	I/O	描述
input_fb	struct __di_fb_t2	输入	当前场的 fb 信息
pre_fb	struct __di_fb_t2	输入	前一场的 fb 信息
next_fb	struct __di_fb_t2	输入	下一场的 fb 信息
output_fb	struct __di_fb_t2	输入	输出场的 fb 信息
field	unsigned int	输入	当前场是 top 场还是 bottom 场的标志
top_field_first	unsigned int	输入	首场是否为 top 场标志

3.2.1.6 __di_fb_t2

描述场的 fb 相关的信息

成员	数据类型	I/O	描述
addr[3]	unsigned long long	输入	存放场的 Y/C 或 Y/U/V 分量数据的物理内存地址
size	__di_rectsz_t	输入	场的像素数据大小
format	__di_pixel_fmt_t	输入	场的格式类型

3.2.1.7 __di_rectsz_t

描述场的 fb 大小相关的信息

成员	数据类型	I/O	描述
width	unsigned int	输入	fb 的宽
height	unsigned int	输入	fb 的高

3.2.1.8 __di_mode_t

定义 di 工作模式和更新方式

成员	数据类型	I/O	描述
di_mode	enum __di_intp_mode_t	输入	di 工作模式
update_mode	enum __di_updmode_t	输入	di 输出的更新方式

3.2.2 模块使用接口说明

di 驱动使用标准的文件 IO 接口：open、close、ioctl。下面描述 ioctl 的命令接口。

3.2.2.1 DI_IOCSTART

synopsis `ioctl(fd, DI_IOCSTART, struct __di_para_t *)`
arguments null
returns
Zero, if successful.
Negative, if there is an error.
description This api set frame fb parameter for di and also trigger di hardware to process fb.

3.2.2.2 DI_IOCSTART2

synopsis `ioctl(fd, DI_IOCSTART, struct __di_para_t2 *)`
arguments null
returns
Zero, if successful.
Negative, if there is an error.
description This api set frame fb parameter for di and also trigger di hardware to process fb.

3.2.2.3 DI_IOCSETMODE

synopsis `ioctl(fd, DI_IOCSETMODE, struct __di_mode_t *)`
arguments null
returns
Zero, if successful.
Negative, if there is an error.
description This api set process mode for di.

3.2.3 驱动调用流程说明

- 1) 使用 open 接口打开 DI 模块，0 使用结束时使用 close 接口关闭 DI 模块
- 2) 拿到当前场、前一场、下一场、输出场等场相关的 fb 信息
- 3) 调用 DI_IOCSETMODE ioctl，设置 di 的工作模式和输出更新方式，可选项
- 4) 调用 DI_IOCSTART/DI_IOCSTART2 ioctl 接口，将 fb 等参数设置到 di 驱中，并触发 di 硬件工作

3.2.4 模式介绍及使用说明

DI200 lowlevel_v1x 硬件版本目前驱动上强制只使用 motion-adaptive 模式，暂不支持模式动态切换。

4 DI110 模块

4.1 模块功能

4.1.1 模块功能介绍

DI (De-Interlace) 即为去隔行，可以把隔行扫描的图像恢复为逐行扫描的图像。该版 DI 硬件提供基础的 DI 功能，为低成本平台提供极具性价比的去隔行解决方案。

4.1.2 模块规格介绍

1. 支持分辨率从 32x32 到 2048x1280
2. 支持场内插值算法
3. 支持边缘/锯齿处理 (Motion-adaptive) 算法，也称为运动检测算法
4. 性能最大支持 1080p @60HZ YUV420 输出 (工作时钟为 600MHz)
5. 支持输入输出的图像像素格式：YUV420 (YV12/NV12/NV21)、YUV422 (YV16/NV16/NV61)
6. 不支持输入输出图像像素格式转换
7. DI110 支持四种模式，分别是 BOB 单帧模式、BOB 双帧模式、MD 单帧模式和 MD 双帧模式

4.1.3 模块配置介绍

4.1.3.1 设备树

```
di: deinterlace@5400000 {
    compatible = "allwinner,sunxi-deinterlace";
    reg = <0x0 0x05400000 0x0 0x0000ffff>;
    interrupts = <GIC_SPI 88 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&ccu CLK_DI>, <&ccu CLK_BUS_DI>, <&ccu CLK_PLL_PERIPH0_2X>;
    clock-names = "clk_di", "clk_bus_di", "pll_periph";
    resets = <&ccu RST_BUS_DI>;
    reset-names = "rst_bus_di";

    assigned-clocks = <&ccu CLK_DI>;
    assigned-clock-parents = <&ccu CLK_PLL_PERIPH0_2X>;
    assigned-clock-rates = <300000000>;

    iommus = <&mmu_aw 4 1>;
    status = "okay";
}
```

}

如上，在 DI 设备树中主要是配置时钟/中断/寄存器/iommu 等硬件资源。

4.1.3.2 menuconfig

4.1.3.2.1 Linux-5.10 以下版本

在命令行中进入 longan 根目录，执行./build.sh menuconfig 进入配置主界面，如下图选上“Support DI V110/120”项即可。

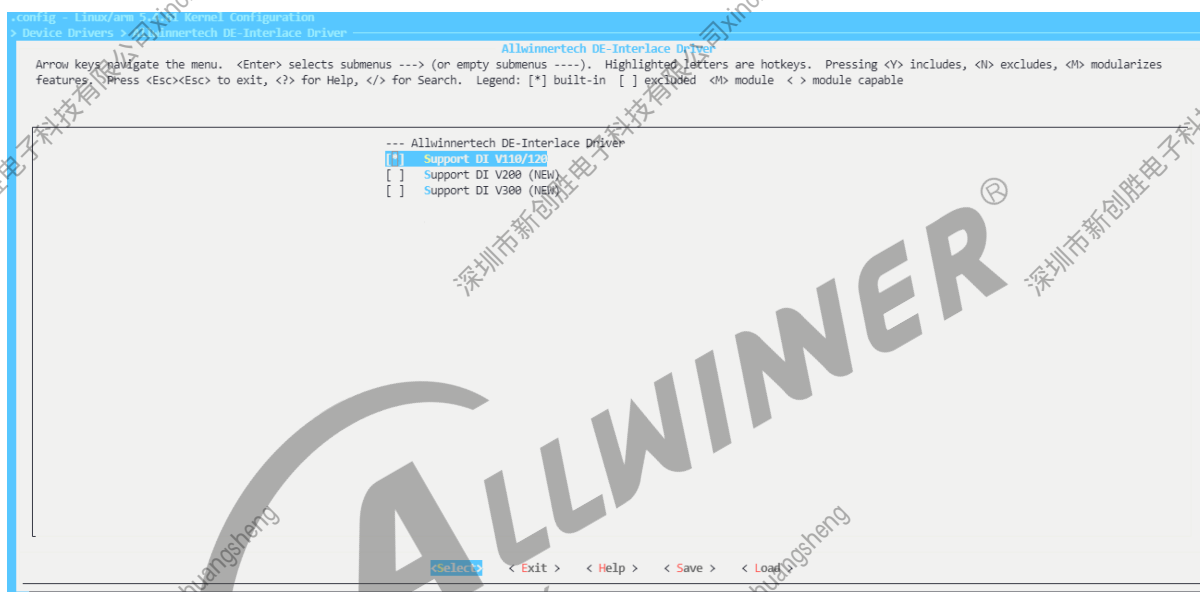


图 4-1: DI menuconfig

4.1.3.2.2 Linux5.10 及其以上版本

在命令行中进入 longan 根目录，执行./build.sh menuconfig 进入配置主界面，如下图选上“Support DI V110/120”项即可。

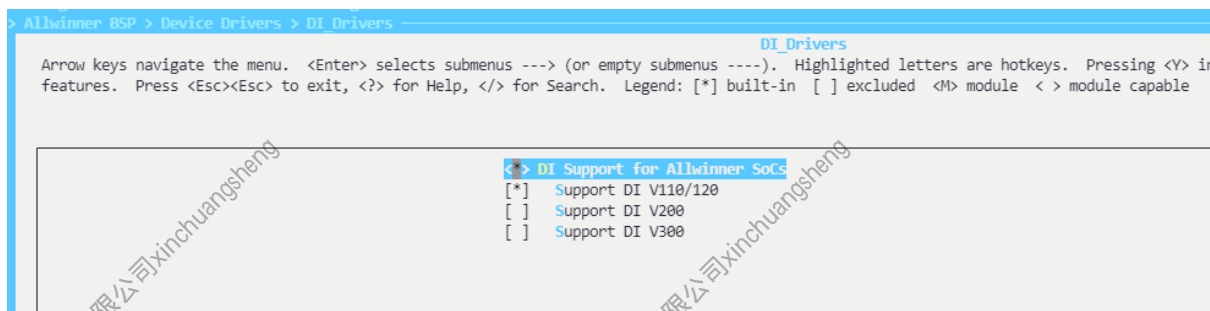


图 4-2: DI menuconfig_5.10 及以上版本

4.1.4 源码结构介绍

```
├── Makefile
├── di110.c
├── di110.h
├── di110_reg.h
├── di_client.c
├── di_client.h
├── di_dev.h
├── di_driver.c
├── di_driver.h
├── di_fops.c
├── di_fops.h
└── sunxi_di.h
```

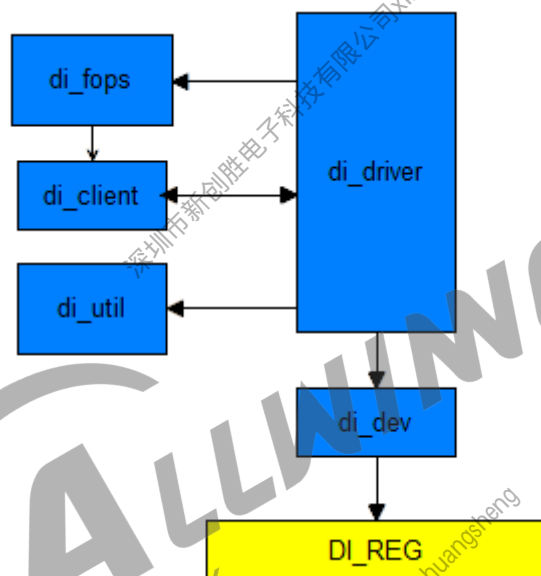


图 4-3: DI 驱动结构

- **di_driver** 模块：负责对接 linux 内核
- **di_fops** 模块：负责管理驱动对用户空间的接口
- **di_client** 模块：负责多用户管理
- **di_util** 模块：提供常用工具函数
- **di_dev** 模块：负责操作 DI 硬件，实现 DI 功能
- **DI_REG**：DI110 硬件寄存器

4.2 模块接口说明

4.2.1 驱动关键结构体说明

4.2.1.1 struct di_version

描述 DI110 驱动的版本，以及 DI 硬件版本

成员	数据类型	I/O	描述
version_major	u32	输出	DI 设备驱动程序编号，默认为 1
version_minor	u32	输出	访问具体的 DI 设备编号，默认为 0
version_patchlevel	u32	输出	DI 驱动的补丁登记，默认为 0
ip_version	u32	输出	DI 硬件版本，DI110 模块时该值为 0x110

4.2.1.2 struct di_mem_arg

描述使用 DI 驱动分配物理内存

成员	数据类型	I/O	描述
size	unsigned int	输入	指定分配物理内存的大小
handle	unsigned int	输出	分配的物理内存在 DI 驱动中的句柄
phys_addr	u64	输出	分配的内存对应的物理地址

4.2.1.3 struct di_timeout_ns

定义等待超时相关的时间，时间单位为 ns。

成员	数据类型	I/O	描述
wait4start	u64	输入	等待另一用户完成处理后获取 di 硬件资源的超时时间。默认值为 3s
wait4finish	u64	输入	等待 di 硬件完成处理的超时时间。默认值为 3s

4.2.1.4 struct di_tnr_mode

描述 tnr 模式的相关定义（仅 DI120 上使用）

成员	数据类型	I/O	描述
mode	u32	输入	去噪模式，值为 DI_TNR_MODE_xxx
level	u32	输入	自适应去噪模式下的强度等级

4.2.1.5 struct di_size

描述图像的分辨率，单位为 pixel

成员	数据类型	I/O	描述
width	u32	输入	宽
height	u32	输入	高

4.2.1.6 struct di_addr

描述保存 YCbCr 数据的各个分量的首物理地址

成员	数据类型	I/O	描述
y_addr	u64	输入	y 分量的首物理地址, 或地址偏移量
cb_addr	u64	输入	cb 分量的首物理地址, 或地址偏移量, 或 0
cr_addr	u64	输入	cr 分量的首物理地址, 或地址偏移量, 或 0

4.2.1.7 struct di_offset

描述保存 YCbCr 数据的各个分量在 dmabuf 中的偏移量

成员	数据类型	I/O	描述
y_offset	u64	输入	y 分量的数据在内存中的地址偏移量
cb_offset	u64	输入	cb 分量的数据在内存中的地址偏移量
cr_offset	u64	输入	cr 分量的数据在内存中的地址偏移量

4.2.1.8 union di_buf

描述 di buf 相关信息

成员	数据类型	I/O	描述
addr	struct di_addr	输入	描述保存 YCbCr 数据的各个分量的首物理地址
offset	struct di_offset	输入	描述保存 YCbCr 数据的各个分量在 dmabuf 中的偏移量

说明

1. 当内存使用物理地址方式时, addr 为 Y/Cb/Cr 分量的首物理地址, offset 结构体为 0
2. 当内存使用 dmabuf 方式时, addr 结构体为 0, offset 为 Y/Cb/Cr 分量的地址偏移量
3. 当像素格式为 uvcombined 时, cr_addr 或者 cr_offset 为 0

4.2.1.9 struct di_fb

描述输入图像的相关信息

成员	数据类型	I/O	描述
dma_buf_fd	s32	输入	dmabuf 文件描述符, 小于 0 为无效值
buf	union di_buf	输入	描述 di buf 相关信息
size	struct di_size	输入	描述图像的分辨率, 单位为 pixel

说明

注意：当 dma_buf_fd 为无效值时, 表示使用物理地址方式; 否则使用 dmabuf 方式

4.2.1.10 struct di_process_fb_arg

描述一次 di process 所需的信息。

成员	数据类型	I/O	描述
is_interlace	u8	输入	1 表示 I 源视频, 0 表示 P 源视频
field_order	u8	输入	1 表示 top_field 的时间戳小, 0 表示 bottom_field 的时间戳小
pixel_format	u32	输入	视频帧的像素格式, 详见 drm_fourcc 定义
size	struct di_size	输入	描述图像的分辨率, 单位为 pixel
output_mode	u8	输入	1 表示双帧输出模式, 0 表示单帧输出模式
di_mode	u8	输入	描述是 MD 模式, 还是 BOB 模式
tnr_mode	struct di_tnr_mode	输入	DI110 不使用该变量, 仅 DI120 上使用
in_fb0	struct di_fb	输入	时域上的第 1 帧输入 fb
in_fb1	struct di_fb	输入	时域上的第 2 帧输入 fb (BOB 模式下 in_fb1 保存输入的帧数据)
in_fb2	struct di_fb	输入	时域上的第 3 帧输入 fb (在 DI110 上没有使用)

成员	数据类型	I/O	描述
out_fb0	struct di_fb	输入	时域上的第 1 帧输出 fb
out_fb1	struct di_fb	输入	时域上的第 2 帧输出 fb
out_tnr_fb	struct di_fb	输入	tnr 输出 fb（在 DI110 上没有使用）

4.2.2 模块使用接口说明

di 驱动使用标准的文件 IO 接口：open、close、ioctl。下面描述 ioctl 的命令接口。

4.2.2.1 DI_IOC_GET_VERSION

synopsis `ioctl(fd, DI_IOC_GET_VERSION, struct di_version *)`
arguments `struct di_version *`
returns
Zero, if successful.
Negative, if there is an error.
description DI driver number and chip version

4.2.2.2 DI_IOC_SET_TIMEOUT

synopsis `ioctl(fd, DI_IOC_SET_TIMEOUT, struct di_timeout_ns *)`
arguments `struct di_timeout_ns *`
returns
Zero, if successful.
Negative, if there is an error.
description This api set timeout values.

4.2.2.3 DI_IOC_PROCESS_FB

synopsis `ioctl(fd, DI_IOC_PROCESS_FB, struct di_process_fb_arg *)`
arguments `struct di_process_fb_arg *`
returns
Zero, if successful.
Negative, if there is an error.
description This api do process fb of DI.

4.2.2.4 DI_IOC_MEM_REQUEST

synopsis `ioctl(fd, DI_IOC_MEM_REQUEST, struct di_mem_arg *)`
arguments `struct di_mem_arg *`
returns
Zero, if successful.
Negative, if there is an error.
description `malloc` a DMA address space buffer

4.2.2.5 DI_IOC_MEM_RELEASE

synopsis `ioctl(fd, DI_IOC_MEM_RELEASE, struct di_mem_arg *)`
arguments `struct di_mem_arg *`
returns
Zero, if successful.
Negative, if there is an error.
description `free` the DMA address space buffer

4.2.3 应用层调用流程说明

4.2.3.1 软件架构

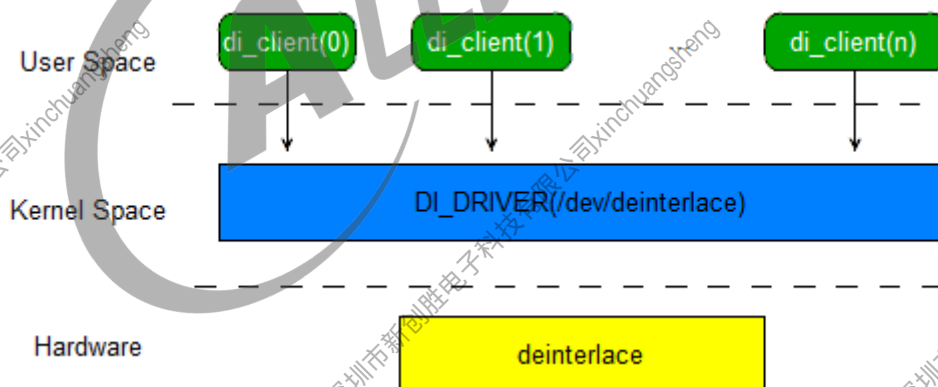


图 4-4: 软件总体架构

支持多用户（多进程）访问模式

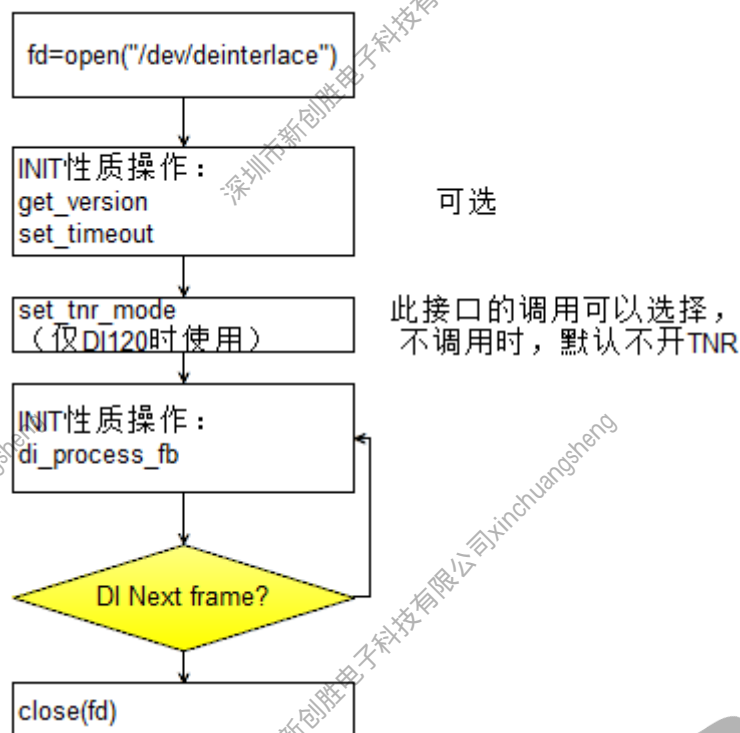


图 4-5: 驱动调用流程图

1. open 打开 DI 模块，使用结束时使用 close 关闭 DI 模块。
2. set_timeout（设置超时等待值）、get_version（获取软硬件版本号），可选项。
3. 循环调用 di_process_fb 进行帧数据的处理。后续章节将详细介绍具体模式下的帧数据的参数要求。
4. 如果需要模式切换（比如从 bob 模式切换到 MD 模式）先 close 后再 open，重新进行设置流程

说明

1. 阻塞式同步方式
2. 对比 DI300 驱动的设计，DI110 驱动减少接口数量，简化上层用户操作 DI 的过程

4.2.3.2 帧处理流程

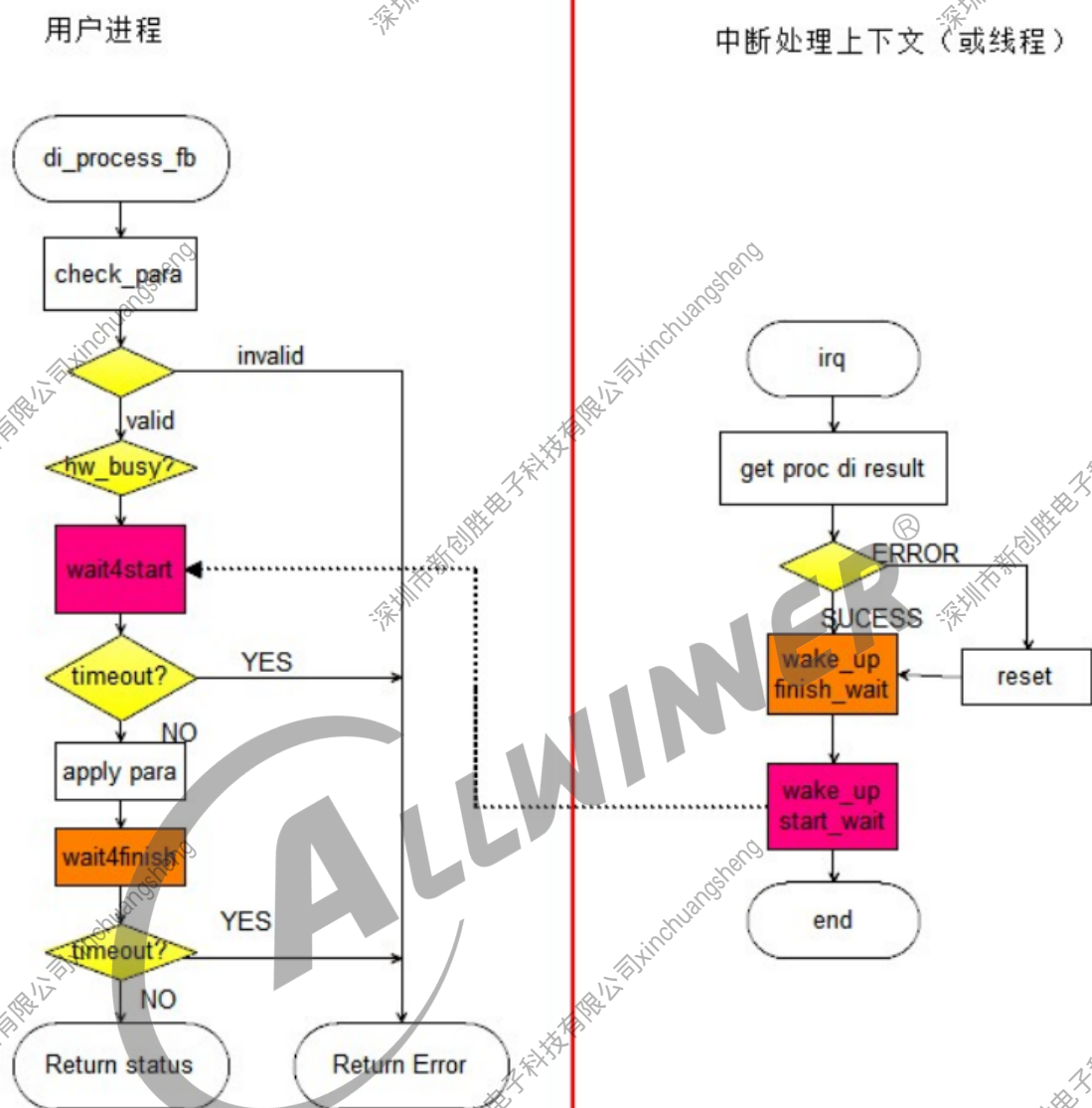


图 4-6: 帧处理流程图

4.2.4 DI 模式介绍及使用说明

DI110 支持四种模式，分别是 BOB 单帧模式、BOB 双帧模式、MD 单帧模式和 MD 双帧模式。

- BOB 模式下，输入只能一帧数据
- MD 模式下，需要输入两帧数据

- BOB 模式和 MD 模式都支持单帧和双帧模式
- 在 BOB 模式下，输入的一帧数据只需要设置给 struct di_process_fb_arg 结构体的 in_fb1 成员，而不是 in_fb0 成员
- 仅有在 MD 模式下时，驱动需要为 DI110 申请和设置一个存储 Motion Flag 信息的 DMA 内存，DI 会将计算出的运动检测信息输出到这块内存中，然后再拿去计算像素的插值

说明

给 Motion Flag 申请 DMA 内存大小的计算公式如下：

双帧时，size = width * height

单帧时，size = width * height / 2

4.2.4.1 BOB 单帧模式

- 使用场内插值算法去隔行
- 输入为一帧，输出也为一帧数据
- 对顶场进行插值，底场由 FB_in 底场原样输出（基场是底场）
- $FB_out = FB_in$ 首场插值 + FB_in 底场
- 单帧模式不考虑 top/bottom field first 参数

4.2.4.1.1 di_process_fb_arg 参数设置

参数	参数值
di_mode	DI_INTP_MODE_BOB
out_frame_mode	DI_DIT_OUT_1FRAME

4.2.4.1.2 数据流基本模型

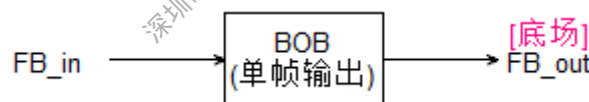


图 4-7: BOB 单帧模式数据流模型

4.2.4.2 BOB 双帧模式

- 使用场内插值算法去隔行
- 输入为一帧，输出为两帧数据
- $FB_out0 = FB_in$ 首场 + FB_in 次场插值
- $FB_out1 = FB_in$ 首场插值 + FB_in 次场

4.2.4.2.1 di_process_fb_arg 参数设置

参数	参数值
di_mode	DI_INTP_MODE_BOB
out_frame_mode	DI_DIT_OUT_2FRAME

4.2.4.2.2 数据流基本模型

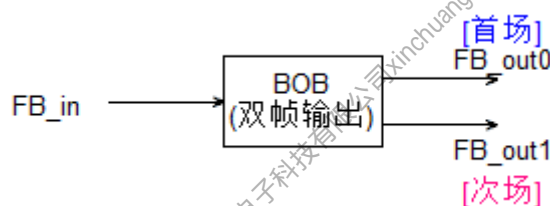


图 4-8: BOB 双帧模式数据流模型

4.2.4.3 MD 单帧模式

- 使用运动检测算法去隔行
- 输入为两帧，输出为一帧数据
- $FB_out = FB_in \text{ 顶场插值} + FB_in \text{ 底场}$
- 单帧模式不考虑 top/bottom field first 参数

4.2.4.3.1 di_process_fb_arg 参数设置

参数	参数值
di_mode	DI_INTP_MODE_MOTION
out_frame_mode	DI_DIT_OUT_1FRAME

4.2.4.3.2 数据流基本模型

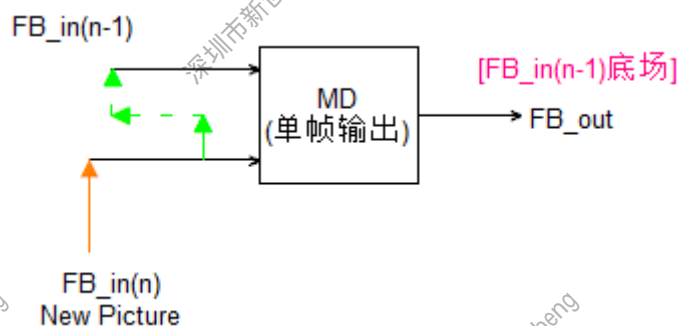


图 4-9: MD 单帧模式数据流模型

说明

MD 模式下的输入规律：

运动检测模式至少有两帧输入，其输入规律为：

第一次 DI: FB_in(0) FB_in(1)

第二次 DI: FB_in(1) FB_in(2)

...

第 N 次 DI: FB_in(n-1) FB_in(n)

4.2.4.4 MD 双帧模式

- 使用运动检测算法去隔行
- 输入为两帧，输出为两帧数据
- $FB_out0 = FB_in0 \text{ 次场} + FB_in0 \text{ 首场插值}$
- $FB_out1 = FB_in1 \text{ 首场} + FB_in1 \text{ 次场插值}$

4.2.4.4.1 di_process_fb_arg 参数设置

参数	参数值
di_mode	DI_INTP_MODE_MOTION
out_frame_mode	DI_DIT_OUT_2FRAME

4.2.4.4.2 数据流基本模型

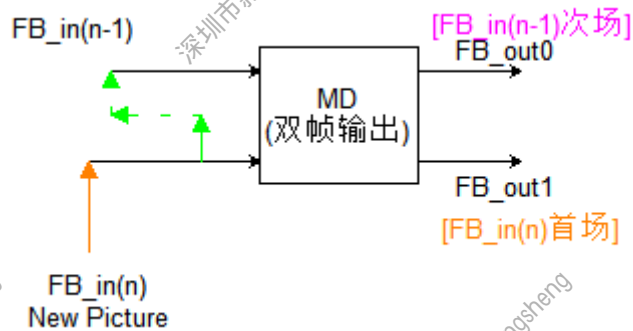


图 4-10: MD 双帧模式数据流模型

说明

则 Top Field First 时:

$FB_out0 = FB_in0 \text{ 底场} + FB_in0 \text{ 顶场插值}$

$FB_out1 = FB_in1 \text{ 顶场} + FB_in0 \text{ 底场插值}$

则 Bottom Field First 时:

$FB_out0 = FB_in0 \text{ 顶场} + FB_in0 \text{ 底场插值}$

$FB_out1 = FB_in1 \text{ 底场} + FB_in0 \text{ 顶场插值}$

5 FAQ

5.1 常用调试方法

通过调整驱动的日志打印等级，实现不同等级日志的输出，通过日志可判断 DI 模块是否工作正常

5.1.1 DI110/DI300 动态调整驱动日志打印等级

模块的日志打印分为两种方式，一种是通过模块的属性节点进行参数设置，另外一种是通过使用 AW 日志基础组件，完成动态控制日志输出的目的。

5.1.1.1 对模块的属性节点进行参数设置

```
cat /sys/class/deinterlace/deinterlace/debug
0: 关闭所有日志打印
1: 打印错误日志
2: 打印关键信息日志
3: 打印更详细的关键信息日志
4: 打印帧处理耗时信息（可观察处理每一帧时，各个阶段的耗时）
5: 打印电影模式的识别情况（DI110没有使用到）
```

打印详细关键信息打印操作如下

```
echo 3 > /sys/class/deinterlace/deinterlace/debug
```

5.1.1.2 使用 AW 日志基础组件

要使用该功能，除了需要内核使能 CONFIG_PRINTK_CALLER 和 CONFIG_DYNAMIC_DEBUG_CORE 配置，另外也需要模块的 Makefile 中定义 DYNAMIC_DEBUG_MODULE 宏。

在确认打开了上述两个配置之后，可以通过读取系统节点，判断 DI 是否使能了动态日志功能

```
/ # cat /proc/dynamic_debug/control | grep di
```

```

/ # cat /proc/dynamic_debug/control | grep di
bsp/drivers/di/drv_div3x/di_client.c:782 [deinterlace]di_client_set_timeout = "[DEBUG]: %s:wait4start=%llu,wait4finish=%llu\n"
bsp/drivers/di/drv_div3x/di_client.c:767 [deinterlace]di_client_set_fmd_enable = "[DEBUG]: %s: fmd_en: en=%d\n"
bsp/drivers/di/drv_div3x/di_client.c:755 [deinterlace]di_client_set_tnr_mode = "[DEBUG]: %s: tnr_mode: mode=%d, level=%d\n"
bsp/drivers/di/drv_div3x/di_client.c:743 [deinterlace]di_client_set_dit_mode = "[DEBUG]: %s: dit_mode: intp_mode=%d, out_frame
bsp/drivers/di/drv_div3x/di_client.c:706 [deinterlace]di_client_set_demo_crop = "[DEBUG]: %s: demo crop: dit: (%u, %u) (%u, %u)
bsp/drivers/di/drv_div3x/di_client.c:680 [deinterlace]di_client_set_video_crop = "[DEBUG]: %s: video_crop: (%u, %u) (%u, %u)\n
bsp/drivers/di/drv_div3x/di_client.c:668 [deinterlace]di_client_set_video_size = "[DEBUG]: %s: video_size[%d x %d]\n"
bsp/drivers/di/drv_div3x/di_client.c:656 [deinterlace]di_client_process_fb = "[DEBUG]: total:%lluus t0~t1:%lluus t1~t2:%l
bsp/drivers/di/drv_div3x/di_client.c:634 [deinterlace]di_client_process_fb = "[DEBUG]: di detect result:interlace, di_mode:%s\
bsp/drivers/di/drv_div3x/di_client.c:623 [deinterlace]di_client_process_fb = "[DEBUG]: di detect result:progressive, di_mode:%
bsp/drivers/di/drv_div3x/di_client.c:423 [deinterlace]di_client_check_fb_arg = "[DEBUG]: out tnr info:format:%s dma_buf_fd:%d \

```

图 5-1: 确认 DI 的动态日志功能是否使能

使用 AW 日志基础组件来进行 DI 的动态日志调试，方法如下：

A. 调整控制台日志等级

```
echo 8 > /proc/sys/kernel/printk
```

B. 仅输出特定 C 文件的 DEBUG 日志

```
echo "file bsp/drivers/di/drv_div3x/di_client.c +p" > /proc/dynamic_debug/control
```

C. 仅输出特定函数的 DEBUG 日志

```
echo "func di_dev_calc_proc_result +p" > /proc/dynamic_debug/control
```

D. 输出 DI 整个模块的 DEBUG 日志

```
echo "module deinterlace +p" > /proc/dynamic_debug/control
```

详细方法可参考 kernel/Documentation/admin-guide/dynamic-debug-howto.rst 文档

5.1.2 DI200 动态调整驱动日志打印等级

de-interlace 驱动通过 sysfs 节点提供了在线调试接口：

```
/sys/module/deinterlace/parameters/debug_mask
```

该节点文件用于控制 DI 驱动的打印等级。缺省值是 0，关闭除出错信息的所有打印。放开全部调试打印信息：

```
echo 0xff > /sys/module/deinterlace/parameters/debug_mask
```

关闭除出错信息的所有打印：

```
echo 0 > /sys/module/deinterlace/parameters/debug_mask
```

还可以对打印信息进行选择性打开。

```
enum {
    DEBUG_INIT = 1U << 0,
    DEBUG_INT = 1U << 1,
    DEBUG_DATA_INFO = 1U << 2,
    DEBUG_SUSPEND = 1U << 3,
    DEBUG_TEST = 1U << 4,
};
```


- DEBUG_INIT 为初始化打印控制开关位，当 debug_mask 的 bit0 为 1 时，此打印打开。
- DEBUG_INT 为中断信息打印控制开关位，当 debug_mask 的 bit1 为 1 时，此打印打开。
- DEBUG_DATA_INFO 为接收发送数据信息打印控制开关位，当 debug_mask 的 bit2 为 1 时，此打印打开。
- DEBUG_TEST 为测试 de-interlace 转换时间的信息打印控制开关位，当 debug_mask 的 bit4 为 1 时，此打印打开。




著作权声明

版权所有 © 2024 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、 全志科技、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标、产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的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所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。