Linux TVD
开发指南

版本号: 1.0
发布日期: 2022.12.22

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.12.22	AWA1442	初始版本

目　录

1 前言

1.1 文档简介

介绍 TVD 模块的使用方法。

1.2 目标读者

TVD 模块驱动及应用层的使用人员。

1.3 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	驱动文件
T113	Linux-5.4	drivers/media/platform/sunxi-tvd/*

2 模块介绍

2.1 模块功能介绍

在 AW 内部，通常把 CVBS IN 模块称为 TVD 或者 TVIN 模块，是一个用于采集模拟 CVBS 视频的硬件模块，可将输入的 CVBS 信号或 YPbPr 信号转换成 YUV 信号。

表 2-1: 平台硬件功能列表

平台名称	通道数	ADC 数	说明
sun8iw20p1	2	1	双通道共用一个 ADC，使用时需切换，因此配置时也认为该平台只有单通道

2.2 相关术语介绍

表 2-2: 术语介绍

术语	说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台。
TVD	TV decoder，全志平台的一个用于采集模拟 CVBS 输入信号的硬件模块

2.3 模块配置介绍

TVD 模块的配置分为两部分，其中关于硬件信息的配置，放在内核目录的 dtsi 文件中，路径为：kernel/linux-5.4/arch/arm64（32 位平台为 arm）/boot/dts/sunxi/CHIP.dtsi(CHIP 为研发代号，如 sun8iw20p1 等)。

关于方案板级相关的配置放在 board.dts 里，路径为：device/config/chips/(芯片名)/configs/(方案名)/linux-5.4/board.dts

2.3.1 Device Tree 配置说明

下面节点用于完成驱动的匹配，完成时钟、中断、tvd top 总体硬件信息的配置。

```
tvd: tvd@01c30000 {
    compatible = "allwinner,sunxi-tvd";
    reg = <0x0 0x01c30000 0x0 0x00010000>;/*tvd_top*/     // 配置tvd top的IO寄存器地址
    clocks = <&clk_tvd_top>;                    // 配置tvd_top的时钟资源
    interrupts = <GIC_SPI 61 IRQ_TYPE_LEVEL_HIGH>;      // 配置tvd top的中断号
    tvd-number = <1>;                           // 配置当前平台硬件支持的tvd通道数
    tvds = <&tvd0>;                             // 配置当前平台用到的通道
    status = "okay";
};
```

下面节点主要配置各 tvd 通道的硬件信息

```
tv0: tvd0@05c01000 {
    compatible = "allwinner,sunxi-tvd0";
    reg = <0x0 0x05c01000 0x0 0x00010000>;              // tvd通道的寄存器IO地址
    interrupts = <GIC_SPI 107 IRQ_TYPE_LEVEL_HIGH>;     // tvd通道的中断号
    clocks = <&ccu CLK_TVD0>， <&ccu CLK_BUS_TVD0>;      // tvd通道的时钟资源
    status = "okay";
};

tvd1: tvd1@05c02000 {
    compatible = "allwinner,sunxi-tvd1";
    reg = <0x0 0x01c32000 0x0 0x00010000>;
    interrupts = <GIC_SPI 108 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&ccu CLK_TVD1>， <&ccu CLK_BUS_TVD1>;
    status = "okay";
};

tvd2: tvd2@05c03000 {
    compatible = "allwinner,sunxi-tvd2";
    reg = <0x0 0x01c33000 0x0 0x00010000>;
    interrupts = <GIC_SPI 109 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&ccu CLK_TVD2>， <&ccu CLK_BUS_TVD2>;
    status = "okay";
};

tvd3: tvd3@05c04000 {
    compatible = "allwinner,sunxi-tvd3";
    reg = <0x0 0x01c34000 0x0 0x00010000>;
    interrupts = <GIC_SPI 110 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&ccu CLK_TVD3>， <&ccu CLK_BUS_TVD3>;
    status = "okay";
};
```

2.3.2 board.dts 配置说明

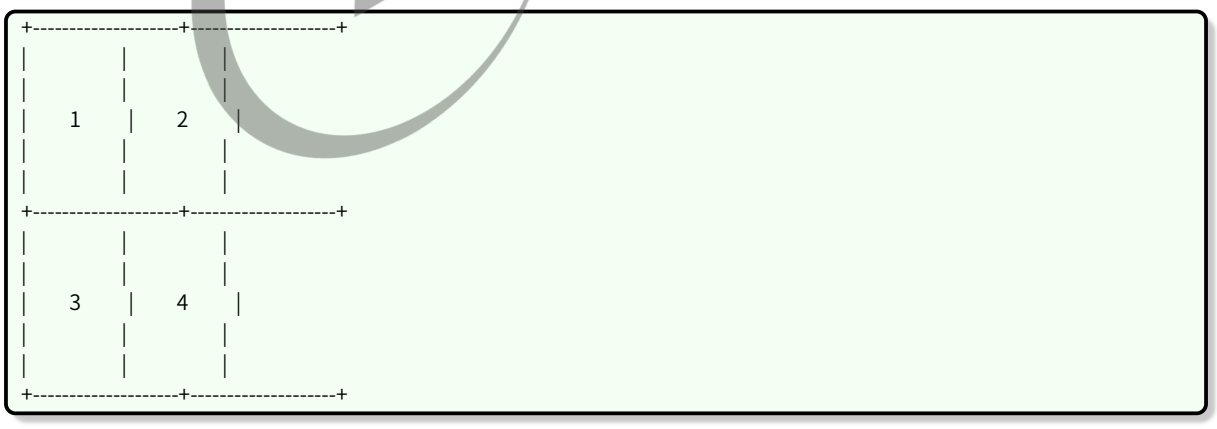
```
&tvd {
    tvd_sw          = <1>;
    tvd_hot_plug    = <1>;
    tvd_interface   = <0>;
    tvd_format      = <0>;
```

```
    tvd_system      = <1>;
    tvd_row         = <2>;
    tvd_column      = <2>;
    tvd_channel0_en = <1>;
    tvd_channel1_en = <2>;
    tvd_channel2_en = <3>;
    tvd_channel3_en = <4>;
    tvd_power0      = "vcc-tvin";
    vcc-tvin-supply = <&reg_cldo1>;
};

&tvd0 {
    tvd_used        = <1>;
    tvd_interface   = <0>;
    agc_auto_enable = <1>;
    agc_manual_value = <64>;
    cagc_enable     = <1>;
    fliter_used     = <1>;
};

&tvd1 {
    tvd_used        = <1>;
    tvd_interface   = <0>;
    agc_auto_enable = <1>;
    agc_manual_value = <64>;
    cagc_enable     = <1>;
    fliter_used     = <1>;
};

&tvd2 {
    tvd_used        = <1>;
    tvd_interface   = <0>;
    agc_auto_enable = <1>;
    agc_manual_value = <64>;
    cagc_enable     = <1>;
    fliter_used     = <1>;
};

&tvd3 {
    tvd_used        = <1>;
    tvd_interface   = <0>;
    agc_auto_enable = <1>;
    agc_manual_value = <64>;
    cagc_enable     = <1>;
    fliter_used     = <1>;
};
```

其中各属性含义如下：

- tvd_used: create tvd device, 0: do not create tvd device; 1: create tvd device
- tvd_hot_plug: 0：关闭热插拔检测功能; 1：打开热插拔检测功能
- agc_auto_enable: 0: agc 手动模式，这个时候需要设置 agc_manual_value; 1: 自动模式，一般情况下请设置为自动模式
- agc_manual_value: agc manual value, default value is 64

- cagc_enable: 0: disable; 1: enable
- fliter_used: 3d 滤波，思路只能使能一路 0: disable; 1: enable
- tvd_sw: tvd 的总开关，必须开启
- tvd_interface: 0: CVBS, 1: YPBPRI, 2:YPBPRP
- tvd_format: 0:TVD_PL_YUV420 , 1: MB_YUV420, 2: TVD_PL_YUV422
- tvd_system: 0:NTSC, 1:PAL
- tvd_row: 多路拼接模式下的行数量：1 到 2
- tvd_column: 多路拼接模式下的列数量：1 到 2
- tvd_row*tvd_column: 这两个数的乘积表示多路拼接模式下总通道数
- tvd_channelx_en: 多路拼接模式下，0 该通道禁止，1 到 4 表示使能在指定位置上，如下图
- tvd_powerX: tvd 的电源名称，X 取值 0 到 1
- xxx-supply: xxx 为上面 tvd_powerX 所填的电源名称，属性需要引用对应的电源句柄
- tvd_gpioX: tvd 的 gpio 配置，X 取值 0 到 1



2.3.3 kernel menuconfig 配置说明

完成平台配置后，如果是 longan 环境，在 longan 目录下输入 (如果是 tina5.0 环境，在 tina 目录下输入)：./build.sh menuconfig，并按下面步骤操作：

```
Device Drivers --->
  <*> Multimedia support --->
    [*] V4L platform devices --->
      <*> sunxi tvd driver
```

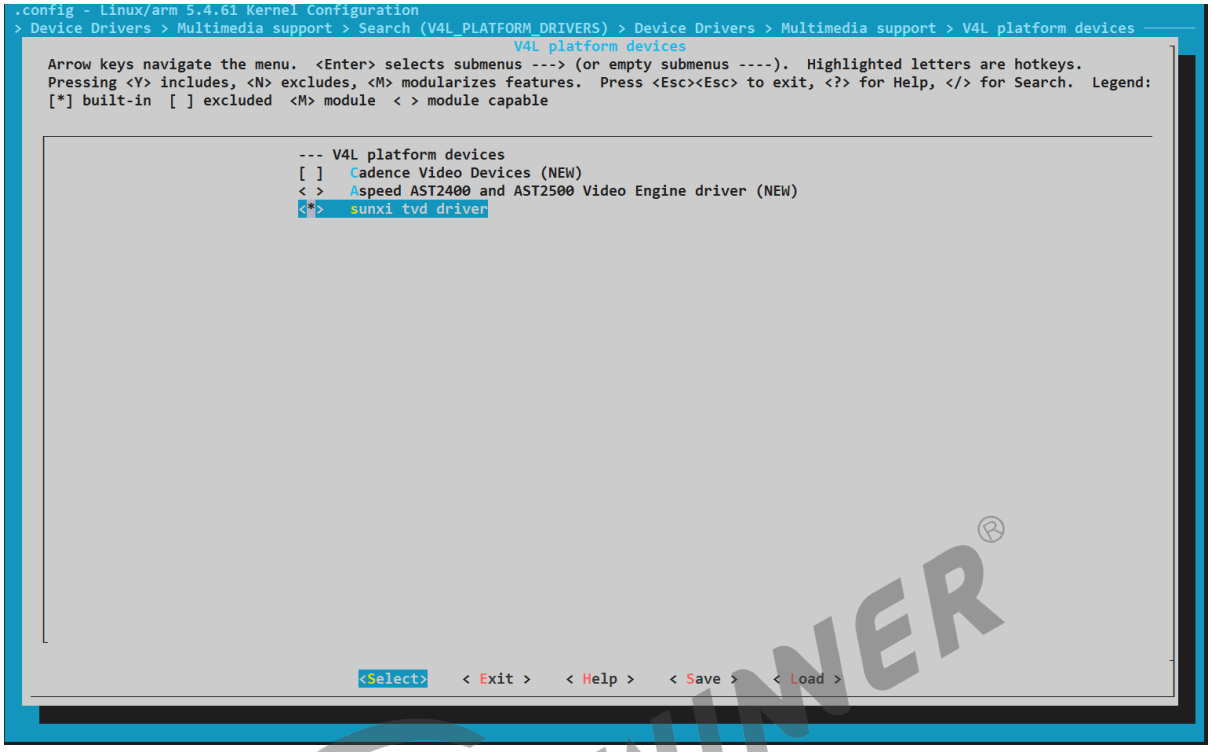


图 2-1: menuconfig 配置

2.4 源码结构介绍

```
drivers/media/platform/sunxi_tvd
├──── tvd.c
├──── bsp_tvd.c
├──── tvd.h
├──── bsp_tvd.h
├──── Makefile
```

- tvd.c: 设备驱动初始化，将设备注册进 V4L2 框架
- bsp_tvd.c: 底层硬件寄存器的操作接口

从上面源码结构可以看到，虽然 TVD 模块也隶属于 CVBS 接口，但是其实这个模块跟 AW 显示框架没什么关系，而是通过多媒体的 V4L2 框架来进行管理。

2.5 驱动框架介绍

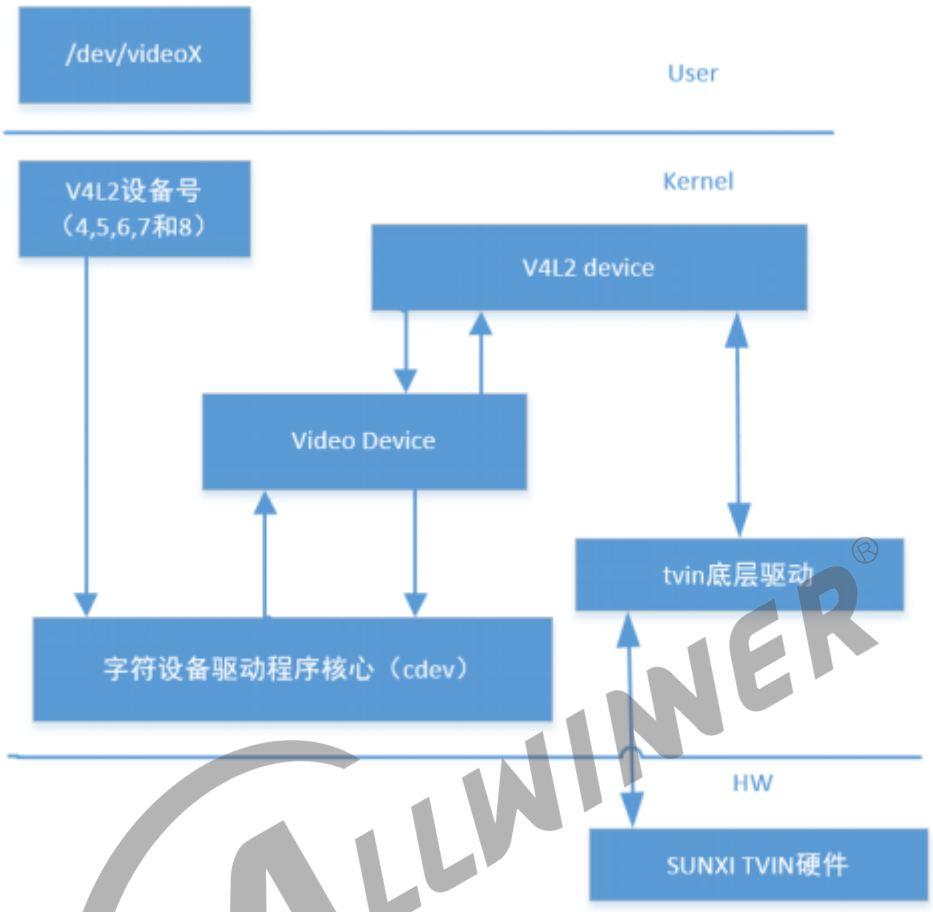


图 2-2: 驱动框架

3 应用流程介绍

tvd 驱动只是负责把 tvd 的硬件描述完成并注册进 V4L2 框架，具体对 tvd 的使用还是放在用户态的应用层。这里归纳关于用户态操作使用 tvd 模块的流程。

3.1 操作单路 tvd

1. 用 open 打开/dev/videox 节点，其中 4 到 7 对应四个通道，video8 对应多路拼接。
2. （仅针对多通道共用 ADC 的平台，如 sun8iw20p1）执行执行 Ioctl:VIDIOC_S_INPUT 完成通道和 ADC 的 mapping，然后执行 Ioctl:VIDIOC_G_INPUT 确认通道和 ADC mapping 正确
3. 执行 ioctl:VIDIOC_G_FMT，用于初始化 format 以及获取当前 tvd 的锁状态
4. 执行 ioctl：VIDIOC_S_FMT，定制 format，包括颜色空间，一般来说 V4L2_PIX_FMT_NV61 要好于 V4L2_PIX_FMT_NV12，多路拼接的时候，在这个 ioctl 中检测锁信号
5. 执行 ioctl：VIDIOC_REQBUFS，通过 V4L2 框架申请 buffer
6. 执行 ioctl：VIDIOC_QUERYBUF，获取 buffer
7. 执行 ioctl：VIDIOC_QBUF，将获取到的 buffer 入列，供 tvd 模块写入数据
8. 执行 ioctl：VIDIOC_STREAMON，使能 tvd 模块工作采集输入数据
9. 通过 poll/select 的方式查询 dev/videox 节点，如有数据更新，执行 ioctl：VIDIOC_DQBUF 取出已经写好数据的 buffer
10. 执行 ioctl：VIDIOC_QBUF，将步骤 9 中已经取出数据的 buffer 重新入列，供 tvd 模块写入数据
11. 循环步骤 9-10，直至需要停止 tvd 模块，执行 ioctl：VIDIOC_STREAMOFF，关闭 tvd 模块
12. 用 close 释放获取的 fd

3.2 操作多路拼接

多路拼接与单路在应用上的区别除了节点不一样之外，多路拼接是打开/dev/video8 之外，就是 VIDIOC_S_FMT 所传参数不同，多路拼接的的时候是通过 v4l2_format.fmt.raw_data 来传递参数，raw_data 是一个数组用于存放多路拼接的配置信息。

raw_data 数组的的定义如下：

```
#define RAW_DATA_INTERFACE  1
#define RAW_DATA_SYSTEM    2
#define RAW_DATA_FORMAT    3
#define RAW_DATA_PIXELFORMAT 4
#define RAW_DATA_ROW      5
#define RAW_DATA_COLUMN    6
#define RAW_DATA_CH0_INDEX  7
#define RAW_DATA_CH1_INDEX  8
#define RAW_DATA_CH2_INDEX  9
#define RAW_DATA_CH3_INDEX  10
#define RAW_DATA_CH0_STATUS 11
#define RAW_DATA_CH1_STATUS 12
#define RAW_DATA_CH2_STATUS 13
#define RAW_DATA_CH3_STATUS 14

raw_data[RAW_DATA_INTERFACE]:  无需设置，默认固定为tvd
raw_data[RAW_DATA_SYSTEM]:    0：NTSC  1:PAL
raw_data[RAW_DATA_FORMAT]:   0：TVD_PL_YUV420 1：TVD_MB_YUV420 2：TVD_PL_YUV422
raw_data[RAW_DATA_PIXELFORMAT]:0：RAW_DATA_FMT_NV12 1：RAW_DATA_FMT_NV21 2：RAW_DATA_FMT_NV16 3
      ：RAW_DATA_FMT_NV61
raw_data[RAW_DATA_ROW]:    行数，值在1~2之间选择
raw_data[RAW_DATA_COLUMN]:    列数，值在1~2之间选择
raw_data[RAW_DATA_CH0_INDEX]:   0：通道关闭 1~4：多路拼接在田字格的位置，见上文
raw_data[RAW_DATA_CH1_INDEX]:   0：通道关闭 1~4：多路拼接在田字格的位置，见上文
raw_data[RAW_DATA_CH2_INDEX]:   0：通道关闭 1~4：多路拼接在田字格的位置，见上文
raw_data[RAW_DATA_CH3_INDEX]:   0：通道关闭 1~4：多路拼接在田字格的位置，见上文
```

参考案例如下：

```
fmt.raw_data[RAW_DATA_PIXELFORMAT] = RAW_DATA_FMT_NV12;
fmt.fmt.raw_data[RAW_DATA_FORMAT] = TVD_PL_YUV420;
fmt.raw_data[RAW_DATA_CH0_INDEX] = 0;
fmt.raw_data[RAW_DATA_CH0_INDEX] = 1;
fmt.raw_data[RAW_DATA_CH0_INDEX] = 2;
fmt.raw_data[RAW_DATA_CH0_INDEX] = 3;
fmt.fmt.raw_data[RAW_DATA_ROW] = 2;
fmt.fmt.raw_data[RAW_DATA_COLUMN] = 2;
```

著作权声明

商标声明

免责声明