AW User Guide Linux DRM CN

版本号: 1.4

发布日期: 2025.07.11

版本历史

版本号	日期	制/修订人	内容描述
1.0	2024.02.28	AWA1639	添加初版
1.1	2024.05.24	AWA1639	适配新版驱动
1.2	2024.11.27	AWA2075	添加私有属性介绍和调试方法
1.2.1	2025.03.21	AWA2075	添加 1919 平台列表
1.3	2025.05.12	AWA1442	支持 T736
1.4	2025.07.11	AWA2447	支持 T153

目　录

1 前言

1.1 文档简介

介绍 Sunxi 平台 DRM/KMS 驱动的概要实现及配置方法。

1.2 目标读者

- DRM 驱动开发人员/维护人员
- DRM 模块的用户态开发人员

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
T527/A523/A527	linux5.10/linux5.15	bsp/drivers/drm/
MR536/T536	linux5.10	bsp/drivers/drm/
T527_AIOT	linux5.15	bsp/drivers/drm/
A733	linux6.6	bsp/drivers/drm/
T736	linux5.15	bsp/drivers/drm/
A537	linux6.6	bsp/drivers/drm/
A333	linux6.6	bsp/drivers/drm/
T153	linux5.10-rt	bsp/drivers/drm/

1.4 相关术语介绍

表 1-2: 术语介绍

术语	解释
DRM	Direct Rendering Manager 直接渲染管理器
KMS	kernel Mode Setting
模式设置	设置显示通路的模式（分辨率，颜色深度，位宽，图层设置等）
libdrm	用于访问操作 DRM 的用户态库
modetest	libdrm 提供的测试程序，可以查询显示设备的信息及进行基本的显示测试
crtc	阴极射线管控制器，drm 中将输入像素根据相应时序发送给下级硬件的硬件抽象
encoder	编码器，作为 crtc 的后级硬件，负责将 crtc 的信号转换给 connector 输出
connector	连接器，显示通路末端显示输出接口的抽象，更多强调的是接口本身，lvds，hdmi，edp 等
plane	图层、平面，即具有合成能力的 crtc 的硬件图层抽象

2 模块介绍

Sunxi DRM 驱动由 Linux kernel DRM 框架（kernel/drivers/gpu/drm/）以及 bsp 仓库下 drivers/drm/目录下的全志平台适配驱动组成，通过 DRM 驱动实现了对全志显示硬件的管理控制。

用户空间可以通过 libdrm 的标准 API 透过 DRM 对全志平台的显示硬件实现对显示通路的全功能管理配置。

2.1 DRM 简介

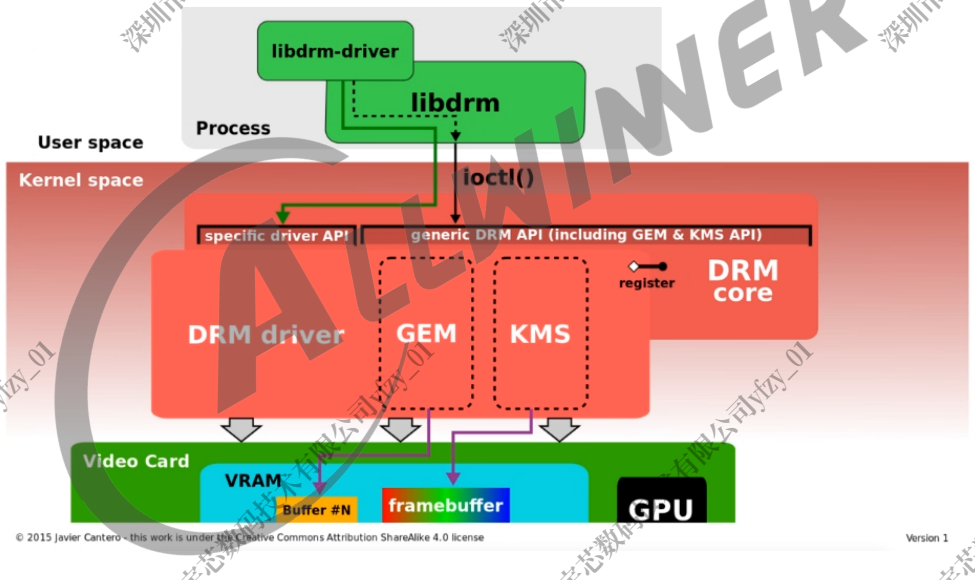

图 2-1: DRM 通路框图

- Linux DRM 包含旨在支持复杂图形设备需求的代码，通常包含适合 3D 图形加速的可编程管道。内核中的图形驱动程序可以利用 DRM 功能来简化内存管理、中断处理和 DMA 等任务，并为应用程序提供统一的接口。
- 直接渲染管理器 (DRM) 是 Linux 内核的一个子系统，负责与现代显卡的 GPU 接口。DRM 公开了一种 API，用户空间程序可以使用该 API 将命令和数据发送到 GPU 并执行配置显示器模式设置等操作。

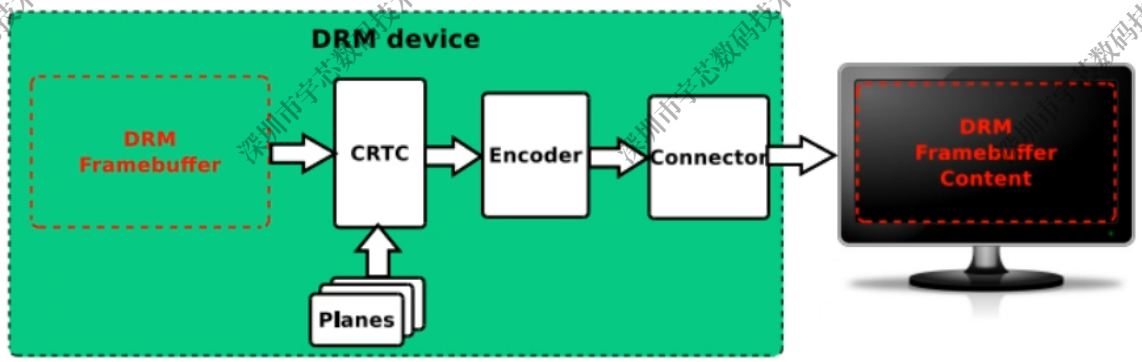

图 2-2: DRM Device 框图

DRM 的几个基础组件关系如上，Framebuffer、CRTC、Encoder、Connector、Plane 都是通过 DRM 对显示通路进行配置操作的关键组件，通过对这些组件的参数进行配置并完成一次 commit，即为一次 modeset，决定了后续整个显示通路的图层缩放叠加混合，末端显示设备的开关、显示模式等显示画面输出相关状态。

2.2 Sunxi 平台 DRM 介绍

DRM 的 AW 平台适配代码放在 bsp/drivers/drm 下，对接完成了 kernel DRM 框架要求驱动所需要实现的接口。

- 支持 edp/hdmi/lvds/rgb/mipi dsi 等显示输出接口
- 支持 bootloader 与内核平滑过渡显示
- 支持 fbdev
- DE 支持 iommu 内存映射
- 支持多图层混合叠加缩放
- 支持 drm atomic 机制

驱动的主要源码文件及说明如下：

```
├── panel                  //屏幕模组驱动
│   ├── dsi-panel-simple.c
│   ├── edp_general_panel.c
│   ├── panel-lvds.c
│   ├── panel-rgb.c
│   ├── panels.c
│   └── sunxi-panel-simple.c
├── phy                    //dsi、lvds、rgb等部分接口模块的phy抽象
│   ├── sunxi_dsi_combophy.c
│   ├── sunxi_dsi_combophy_reg.c
├── sunxi_device
│   ├── hardware
│   │   ├── lowlevel_de    //de底层硬件相关
```

```
|   |   ├── lowlevel_edp      //edp/dp底层硬件相关
|   |   ├── lowlevel_hdmi20   //hdmi底层硬件相关
|   |   ├── lowlevel_lcd      //dsi、lvds、rgb等接口模块相关
|   |   └── lowlevel_tcon     //tcon-tv底层硬件相关
|   ├── sunxi_edp.c          //edp/dp中间抽象层
|   ├── sunxi_hdmi.c         //hdmi中间抽象层
|   ├── sunxi_tcon.c         //tcon驱动顶层实现
|   ├── sunxi_tcon_top.c     //tcon_top驱动顶层实现
├── sunxi_drm_crtc.c         //crtc/plane对接，对下调用lowlevel_de
├── sunxi_drm_drv.c          //drm顶层master驱动
├── sunxi_drm_dsi.c          //dsi接口对接drm connector及encoder
├── sunxi_drm_edp.c          //edp接口对接drm connector及encoder
├── sunxi_drm_hdmi.c         //hdmi接口对接drm connector及encoder
├── sunxi_drm_lvds.c         //lvds接口对接drm connector及encoder
├── sunxi_drm_rgb.c          //rgb接口对接drm connector及encoder
├── sunxi_fbdev_core.c       //fbdev实现
└── sunxi_fbdev_platform.c   //fbdev实现
```

2.3 硬件模块介绍

AW 平台显示相关硬件 IP 模块主要包括 de、tcon top（又称 disp sys）、combp phy、tcon、hdmi、edp/dp、DSI、CVBS、VGA 等，如下框图。

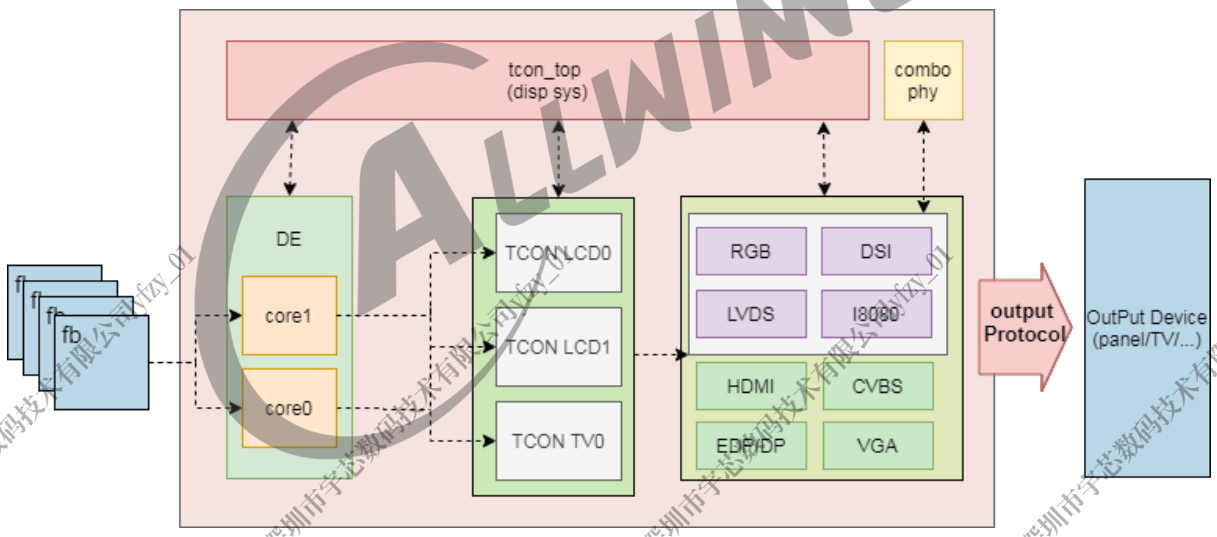

图 2-3: 硬件概览

内存中的输入 buffer 经过 DE 处理后，连接到 tcon，经过输出协议 IP 模块处理最终通过 SOC 的外接引脚输出相应的显示输出协议信号。

一路显示输出通常始于 DE 的一个 core 或者一路 disp output，DE 对内存中的多个图像数据（fb）做混合裁剪及画质调整等处理后，信号输出连接到一路 tcon（timing controller）。

tcon 分为 tcon_tv 以及 tcon_lcd，tcon 后端一般为接口协议模块的具体 IP，但也有部分接口（如 RGB、LVDS 等）直接由 tcon_lcd 模块内部实现，不需要后接额外的协议 IP 模块。

协议 IP 模块一般都有自己的 phy 模块，图中省略未画出，但值得注意的是 combo phy 也是一个独立的硬件 phy 模块，但其同时为多种接口（如 RGB、LVDS 等）提供一些功能配置及实现。

在多数情况下，一路 DE core 后接一路 tcon，再根据实际后端可能再有一路协议 IP 模块来完成信号输出。

但在某些情况下也可能后端协议模块需要通过两路 tcon 来完成通路数据传递（如 dual dsi），此时后端的协议 IP 模块可能也需要两路同时输出。

DE 在 drm 中被抽象为 crtc，DE 具有的硬件图层合成能力，每个 blending channel 被抽象为一个 plane，各种硬件输出接口则被抽象为 connector，这些硬件的操作方法均通过 DRM 的标准的 API 提供。

用户空间一般借助 libdrm 简化对 DRM 框架的 api 调用，完成对显示输出管理，buffer 申请、释放、映射、合成等操作。

技巧

不同的显示输出接口协议硬件上固定使用tcon tv或者tcon lcd，tcon可以理解为是接口协议模块与DE沟通时序的桥梁模块。

技巧

支持多显时，每一路的DE输出所提供的功能会有所差异，但一般支持连接到任意输出接口。

3 配置集成方法

3.1 menuconfig

3.1.1 必选配置

在 longan 目录下执行./build.sh menuconfig , 首先依次进入 Device Drivers > Graphics support，选上 Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) 后保存退出。

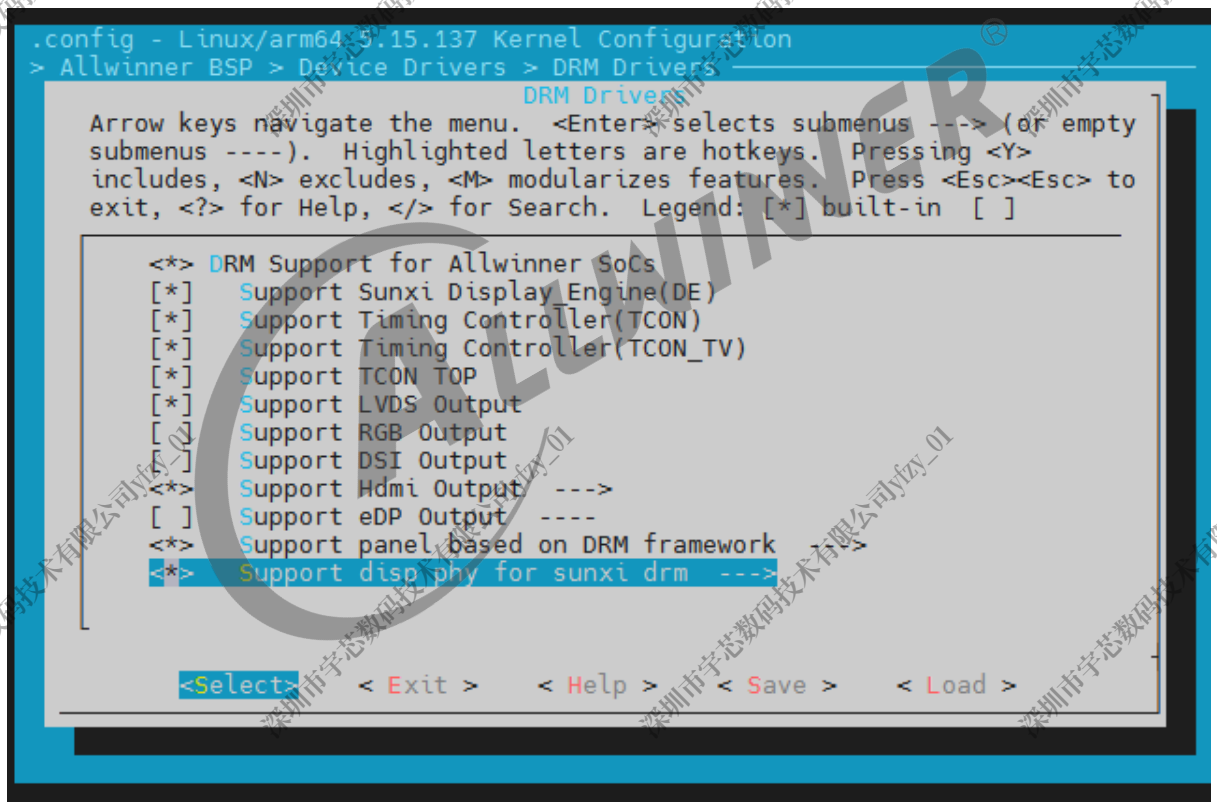

图 3-1: sunxi-drm menuconfig

再执行一次./build.sh menuconfig 依次进入 Allwinner BSP > Device Drivers > DRM Drivers，可见上图的配置页面，其中:

Support Sunxi Display_Engine(DE)、Support Timing Contoller(Tcon)、Support TCON TOP 为必选，

DRM Support for Allwinner SoCs，根据需要可以编译为 build-in 或 module。

3.1.2 可选输出接口配置

hdmi/edp/LVDS/RGB/DSI 根据需要选择，且 hdmi、edp 选上时其下级菜单也需要根据实际打开配置，

需要使用屏幕模组驱动的接口，也需要选上 Support panel based on DRM framework，并在下级菜单中打开对应的屏幕模组驱动。

Support disp phy for sunxi drm 是指 combo phy 驱动，当所使用的接口依赖 combp phy 时，需要选上，如果不确定，请选上，注意下级菜单中的配置也需要选上。

3.1.3 menuconfig 注意事项

另外需要注意的是旧版显示驱动配置项 AW_DISP2，即

```
Allwinner BSP->Device Drivers->Video Drivers->DISP Driver Support(sunxi-disp2)
```

必须配置为关闭。

此外 lcd/dp/edp 等依赖屏幕面板的接口一般需要使用 pwm 背光，对应配置项为 BACKLIGHT_PWM，即

```
Device Drivers->Graphics support->Backlight & LCD device support->Lowlevel Backlight controls->Generic PWM based
    Backlight Driver
```

必须选上。

3.2 board.dst 配置

board.dts 中需要为板级相关的设备进行配置，主要包括显示输出接口相关的设备，即在 board.dts 中需要配置输出接口是否使用，屏幕面板设备的驱动及参数等。

以下为某平台的 board.dts drm 显示相关的节点：

```
{
    lvds_panel: lvds_panel@0 {
        compatible = "sunxi-lvds";
        status = "okay";
        power0-supply = <&reg_dc1sw1>;
        power1-supply = <&reg_bldo2>;
        backlight = <&backlight0>;
        bus-format = <MEDIA_BUS_FMT_RGB888_1X7X4_SPWG>;
        enable0-gpios = <&pio PD 21 GPIO_ACTIVE_HIGH>;
        enable1-gpios = <&pio PD 22 GPIO_ACTIVE_HIGH>;
        display-timings {
            native-mode = <&lvds0_timing0>;
            lvds0_timing0: timing0 {
                clock-frequency = <74871600>;
```

```
            hback-porch = <70>;
            hactive = <1280>;
            hfront-porch = <83>;
            hsync-len = <18>;
            vback-porch = <13>;
            vactive = <800>;
            vfront-porch = <37>;
            vsync-len = <10>;
          };
        };
        port {
          lvds_panel_in: endpoint {
            remote-endpoint = <&lvds_panel_out>;
          };
        };
      };

      backlight0: backlight0 {
        compatible = "pwm-backlight";
        status = "okay";
        brightness-levels = <
           0  1  2  3  4  5  6  7
           8  9 10 11 12 13 14 15
          16 17 18 19 20 21 22 23
          24 25 26 27 28 29 30 31
          32 33 34 35 36 37 38 39
          40 41 42 43 44 45 46 47
          48 49 50 51 52 53 54 55
          56 57 58 59 60 61 62 63
          64 65 66 67 68 69 70 71
          72 73 74 75 76 77 78 79
          80 81 82 83 84 85 86 87
          88 89 90 91 92 93 94 95
          96 97 98 99 100 101 102 103
         104 105 106 107 108 109 110 111
         112 113 114 115 116 117 118 119
         120 121 122 123 124 125 126 127
         128 129 130 131 132 133 134 135
         136 137 138 139 140 141 142 143
         144 145 146 147 148 149 150 151
         152 153 154 155 156 157 158 159
         160 161 162 163 164 165 166 167
         168 169 170 171 172 173 174 175
         176 177 178 179 180 181 182 183
         184 185 186 187 188 189 190 191
         192 193 194 195 196 197 198 199
         200 201 202 203 204 205 206 207
         208 209 210 211 212 213 214 215
         216 217 218 219 220 221 222 223
         224 225 226 227 228 229 230 231
         232 233 234 235 236 237 238 239
         240 241 242 243 244 245 246 247
         248 249 250 251 252 253 254 255>;
        default-brightness-level = <200>;
        enable-gpios = <&pio PI 2 GPIO_ACTIVE_HIGH>;
        pwms = <&pwm0 4 50000 0>;
      };
    };

    &de {
```

```
    chn_cfg_mode = <3>;
    status = "okay";
};

&vo0 {
    status = "okay";
};

&vo1 {
    status = "okay";
};

&lvds0 {
    status = "okay";
    dual-channel = <0>;
    pinctrl-0 = <&lvds0_pins_a>;
    pinctrl-1 = <&lvds0_pins_b>;
    pinctrl-names = "active","sleep";
    ports {
        port@1 {
            reg = <1>;
            lvds_panel_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&lvds_panel_in>;
            };
        };
    };
};

&dsi0combophy {
    status = "okay";
};

&dsi1combophy {
    status = "okay";
};

&drm_edp {
    status = "disabled";

    edp_ssc_en = <0>;
    edp_ssc_mode = <0>;
    edp_psr_support = <0>;
    edp_colordepth = <8>; /* 6/8/10/12/16 */
    edp_color_fmt = <0>; /* 0:RGB  1: YUV444  2: YUV422 */

    lane1_sw = <0>;
    lane1_pre = <0>;
    lane2_sw = <0>;
    lane2_pre = <0>;
    lane3_sw = <0>;
    lane3_pre = <0>;
    efficient_training = <0>;

    sink_capacity_prefer = <1>;
    edid_timings_prefer = <1>;
    timings_fixed = <1>;

    vcc-edp-supply = <&reg_bldo3>;
    vdd-edp-supply = <&reg_dcdc2>;
```

```
    panel = <&edp_panel>;
    ports {
      edp_out: port@1 {
        edp_panel_out: endpoint@0 {
          reg = <0>;
          remote-endpoint = <&edp_panel_in>;
        };
      };
    };
};

&hdmi {
    hdmi_used = <1>;
    bldo3-supply = <&reg_bldo3>;
    hdmi_power0 = "bldo3";
    hdmi_power_cnt = <1>;
    hdmi_hdcp_enable = <1>;
    hdmi_hdcp22_enable = <0>;
    hdmi_cts_compatibility = <0>;
    hdmi_cec_support = <1>;
    hdmi_cec_super_standby = <1>;
    hdmi_skip_bootedid = <1>;

    ddc_en_io_ctrl = <0>;
    power_io_ctrl = <0>;

    status = "okay";
};

&dsi0 {
    status = "okay";
    pinctrl-0 = <&dsi0_4lane_pins_a>;
    pinctrl-1 = <&dsi0_4lane_pins_b>;
    pinctrl-names = "active","sleep";

    panel: panel@0 {
      compatible = "panel-dsi";
      status = "okay";
      reg = <0>;
      power0-supply = <&reg_dc1sw1>;
      power1-supply = <&reg_bldo2>;

      enable0-gpios = <&pio PD 22 GPIO_ACTIVE_HIGH>; //reset
      enable1-gpios = <&pio PH 9 GPIO_ACTIVE_HIGH>; //reset
      enable2-gpios = <&pio PH 10 GPIO_ACTIVE_HIGH>; //reset
      reset-gpios = <&pio PD 21 GPIO_ACTIVE_HIGH>; //reset
    //  backlight = <&backlight0>;
    //  dsi,flags = <(MIPI_DSI_MODE_VIDEO | MIPI_DSI_SLAVE_MODE)>;
      dsi,flags = <(MIPI_DSI_MODE_VIDEO)>;
      dsi,lanes = <4>;
      dsi,format = <0>;
      panel-init-sequence = [
        15 00 02 00 00
        15 00 02 FA 5A
        15 00 02 00 00
        39 00 04 FF 82 05 01
        15 32 02 29 00
        15 00 02 35 00
      ];
```

```
        panel-exit-sequence = [
            05 00 01 28
            05 78 01 10
        ];

        display-timings {
            native-mode = <&timing0>;
            timing0: timing0 {
                clock-frequency = <171993120>;
                hback-porch = <26>;
                hactive = <1200>;
                hfront-porch = <46>;
                hsync-len = <10>;
                vback-porch = <20>;
                vactive = <2000>;
                vfront-porch = <212>;
                vsync-len = <4>;
            };
        };
    };
};
```

当前 DRM 驱动支持 hdmi、edp、dp、dsi、lvds、rgb 接口输出，根据实际硬件平台不同，支持的接口也有所差异。

一些一般不需要修改的 dts 信息均存放在 bsp/configs/linux-x.y/sunXXiwXpX.dtsi 中，同时参照此部分 dtsi 显示相关的设备树节点会对 sunxi-drm 的设备树有更深入理解。

3.2.1 dts graphic 说明

de、tcon、输出接口、屏驱动的设备树节点直接会使用 port 以及 endpoint 的方式连接起来，用来表示这些设备实际支持的连接状态。

这些设备会包含 ports 或者 port 节点，节点的名字会带有 xxx_in 或者 xxx_out，说明这是该设备的一个输出或者输入口。

每一个 port 下会包含若干个 endpoint，指明该设备与其他设备可能的连接关系。

每个 endpoint 下包含一个 remote-endpoint 的属性，其值为该口所连接到的设备的 endpoint 的引用，指明该设备与另一设备在硬件上允许的输入输出连接关系

以下为一个简单的简化设备树节点，de 只支持一路输出，且只有一路 tcon，该 tcon 支持 lvds 及 dsi 输出,lvds 和 dsi 后分别接一个屏。

```
de: de@5000000 {
    ports {
        disp0_out_tcon0: endpoint@0 {
            remote-endpoint = <&tcon0_in_disp0>;
        };
    };
};

lcd0: tcon0@5501000 {
```

```
    ports {
      tcon0_in: port@0 {
        tcon0_in_disp0: endpoint@0 {
          remote-endpoint = <&disp0_out_tcon0>;
        };
      };
      tcon0_out: port@1 {
        tcon0_out_lvds0: endpoint@0 {
          remote-endpoint = <&lvds0_in_tcon0>;
        };
        tcon0_out_dsi0: endpoint@1 {
          remote-endpoint = <&dsi0_in_tcon0>;
        };
      };
    };
};

lvds0: lvds0@0001000 {
    ports {
      lvds0_in: port@0 {
        lvds0_in_tcon0: endpoint@0 {
          remote-endpoint = <&tcon0_out_lvds0>;
        };
      };
      lvds0_out: port@1 {
        lvds0_out_panel: endpoint@0 {
        remote-endpoint = <&lvds_panel_in>;
      };
    };
};

lvds_panel: lvds_panel@0 {
    port {
      lvds_panel_in: endpoint {
        remote-endpoint = <&lvds0_out_panel>;
      };
    };
};

dsi0: dsi0@5506000 {
    ports {
      dsi0_in: port@0 {
        dsi0_in_tcon0: endpoint@0 {
          remote-endpoint = <&tcon0_out_lvds0>;
        };
      dsi0_out: port@1 {
        dsi_out_panel: endpoint@0 {
        remote-endpoint = <&dsi_panel_in>;
      };
    };

    panel: panel@0 {
      port {
        dsi_panel_in: endpoint {
          remote-endpoint = <&dsi_out_panel>;
        };
    };
};
```

上图的连接示意图如下：

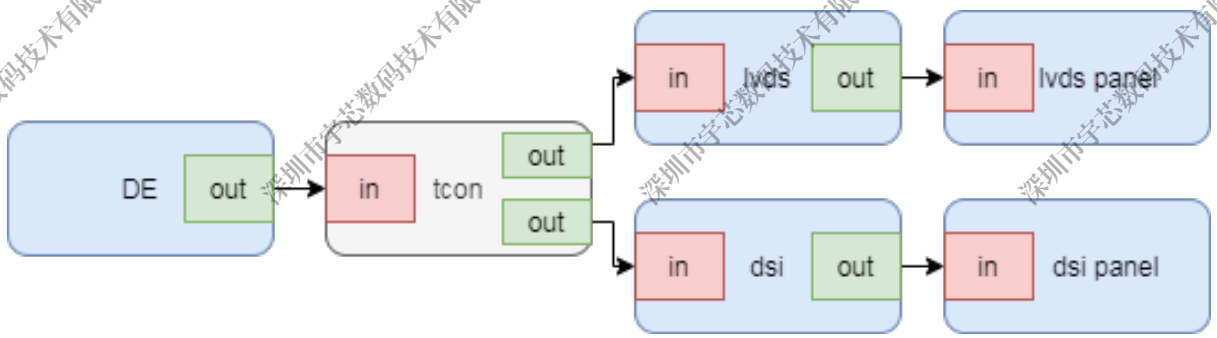

图 3-2: 连接示意图

de 作为显示通路的开始，必然只有输出的 port，port 的数量就是所支持的显示输出的数量。de 的每个 port 下的节点就是这路 de 输出允许连接的 tcon。每一路 tcon 至少包含连接到 de 的输入 port，以及连接到后端显示接口的输出 port。需要连接屏幕模组的接口的设备树节点必然有输出 port 连接到屏幕模组所在节点。dsi 屏幕模组实际是 dsi 总线下的子设备，所以 dsi 屏幕模组的设备树节点从属于 dsi 模块的设备树节点。

3.2.2 详细说明

tcon_top(即 vo,video out) 为 tcon 的额外控制模块，部分 IC DE 也依赖 tcon_top，而 tcon 实际可区分为 tcon_tv 以及 tcon_lcd 两个不同的 ip 模块实现，

如果存在 tcon_top，则每个 tcon 设备必然配套一个 tcon_top 设备，当需要手动配置使能某种输出接口（LVDS/RGB/HDMI 等）时，

切记要保证对应的 tcon 以及对应 tcon 的 tcon_top 的节点的 status 也为 okay。在 sunXXiwX.dtsi 的对应输出接口的节点的 remote-endpoint 可以索引到其所依赖的 tcon 设备。

tcon_lcd 单独可实现 rgb、lvds 两种接口的输出，dsi 接口输出则需要 tcon-lcd 后接 dsi 模块 ip 完成支持，tcon_tv 用于作为 hdmi、edp、dp 输出通路的一部分，后级还需要加上对应输出接口的 ip 模块。

对于依赖 tcon_lcd 输出的输出接口，有可能还会依赖 combo phy 这一硬件模块，是否依赖能在 sunXXiwX.dtsi 对应节点中找到 phys 节点，如果有，则说明有依赖。

当前支持的所有输出通路如下:

- edp/dp 接口: DE->tcon_tv->dp(edp)->panel，根据实际同时可能需要 tcon_top。
- hdmi 接口: DE->tcon_tv->hdmi，根据实际同时可能需要 tcon_top。
- dsi 接口: DE->tcon_lcd->dsi->panel，根据实际同时可能需要 tcon_top，需要 combo phy。
- lvds 接口: DE->tcon_lcd->panel，根据实际同时可能需要 tcon_top，可能需要 combo phy。
- rgb 接口：DE->tcon_lcd->panel，根据实际同时可能需要 tcon_top，可能需要 combo phy。

总而言之，整个 SOC 的显示的通路结构通过 sunXXiwX.dtsi 中相关设备的 ports/endpoint 结构以

图形的形式组织起来，其依赖的设备也以引用的方式包含在对应的设备树节点中，dts 展示出硬件允许的所有可能的显示通路的连接关系，当需要支持某种接口时，需要确保整个通路上的设备都是 okay 状态，并且对应驱动的内核 menuconfig 配置项打开硬件才能正常初始化。

3.3 fbdev 相关配置

drm 可以认为是 fbdev 的下一代显示驱动框架，sunxi-drm 同时也提供了对 fbdev 的兼容处理，支持两者同时使用。

内核的 menuconfig 配置项 DRM_FBDEV_EMULATION，即

```
Device Drivers > Graphics support > Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) > Enable legacy
    fbdev support for your modesetting driver
```

控制是否同时使用 fbdev 兼容，平滑显示 logo 需要与 fbdev 配套使用，即如需使用平滑显示 logo，则需要使能该项配置。

fbdev 默认实现为双 buffer。

fbdev 默认使用 DE 的 channel 1，如果用户态后续需要使用 fbdev，那么通过 drm 送显应避开该 channel 的使用，以达成 drm 与 fbdev 同时使用的目的。

说明

如果用户态没有进程在使用drm驱动，或者在使用完毕后调用了DRM_IOCTL_DROP_MASTER，那么fbdev的刷新将自动完成。否则，fbdev将随着用户态对drm操作的其他plane的送显一同刷新。

4 私有属性

sunxi_drm 驱动，主要使用 atomic commit 配置显示驱动，这里主要介绍一下全志平台相关的一些私有属性，部分 blob 结构，通过 include/video/sunxi_drm.h 透出公共结构体供用户态配置。

主要包括：

4.1 crtc

名字	类型	简介
FEATURE	immutable blob	绑定在 crtc，描述 crtc 硬件特性
BACKEND_DATA	blob	配置后级输出的画质模块，对应 struct de_backend_data
SUNXI_CTM	blob	配置全局的 color materx，3 x 4，对比 drm 多了常数项，对应 struct de_color_ctm
FRAME_RATE_CHANGE	range	高刷相关，用来表示这次 modeset 与改变帧率相关

4.2 plane

名字	类型	简介
FB_ID1~3	object	framebuffer obj id
SRC_X1 ~3	signed range	src x 坐标
SRC_Y1~3	signed range	src y 坐标
SRC_W1~3	signed range	src 的 width
SRC_H1~3	signed range	src 的 height
CRTC_X1~3	signed range	crtc x 坐标
CRTC_Y1~3	signed range	crtc y 坐标

名字	类型	简介
CRTC_W1~3	signed range	crtc 的 width
CRTC_H1~3	signed range	crtc 的 height
COLOR0~3	signed range	配置纯色图片
alpha0 ~ 3	range	配置 global alpha
pixel blend mode0 ~ 3	enum	配置混合模式
COLOR_SPACE	signed range	配置图层的颜色空间
COLOR_RANGE	signed range	配置图层的颜色范围
EOTF	signed range	配置图层的 eotf
compressed_image_crop	range	afbc 图层 crop 信息
FRONTEND_DATA	blob	配置前级输出（视频）的画质模块，对应 struct de_frontend_data
FEATURE	immutable blob	plane crtc 硬件特性

技巧

全志平台部分 palne 硬件对应可以配置 4 个 layer，这些图层只能简单叠加覆盖，且格式、缩放系数等需要一致，所以部分属性会透出 4 个

5 调试方法

5.1 drm-log

- echo 0x1cf > /sys/module/drm/parameters/debug

打开除了 drm vblank 相关的 log 打印，可以观察 drm 调用过程是否正常/报错信息

相关 mask bit 的含义：

```
enum drm_debug_category {
    /**
     * @DRM_UT_CORE: Used in the generic drm code: drm_ioctl.c, drm_mm.c,
     * drm_memory.c, ...
     */
    DRM_UT_CORE     = 0x01,
    /**
     * @DRM_UT_DRIVER: Used in the vendor specific part of the driver: i915,
     * radeon, ... macro.
     */
    DRM_UT_DRIVER       = 0x02,
    /**
     * @DRM_UT_KMS: Used in the modesetting code.
     */
    DRM_UT_KMS      = 0x04,
    /**
     * @DRM_UT_PRIME: Used in the prime code.
     */
    DRM_UT_PRIME        = 0x08,
    /**
     * @DRM_UT_ATOMIC: Used in the atomic code.
     */
    DRM_UT_ATOMIC       = 0x10,
    /**
     * @DRM_UT_VBL: Used for verbose debug message in the vblank code.
     */
    DRM_UT_VBL      = 0x20,
    /**
     * @DRM_UT_STATE: Used for verbose atomic state debugging.
     */
    DRM_UT_STATE        = 0x40,
    /**
     * @DRM_UT_LEASE: Used in the lease code.
     */
    DRM_UT_LEASE        = 0x80,
    /**
     * @DRM_UT_DP: Used in the DP code.
     */
    DRM_UT_DP       = 0x100,
    /**
```

```
    * @DRM_UT_DRMRES: Used in the drm managed resources code.
    */
   DRM_UT_DRMRES      = 0x200,
};
```

5.2 sunxi drm status

```
cat /proc/sunxi-drm/status
```

用来查看 crtc plane 的配置状态，整体的通路 crtc: plane（module）+ plane（module）+ … -> blender（module）

```
plane[93]: plane-0-vch0(0) enable
      crtc=DE-0
      rotation=1
      normalized-zpos=0
            layer_id | fb_id |     src-pos       |     crtc-pos      | blend_mode |  alpha |  color
            ---------+-------+-------------------+-------------------+------------+--------+----------
              0   |  164  | 1280x 800+   0+   0 | 1280x 800+   0+   0 |   premult  |   255  | 0x00000000

      ovl@  101000: disable

      afbd@  105000: disable

      tfbd@  105400: enable
            format: AB24 little-endian (0x34324241) lossy: 0 layout: 16x4
            in(1280x 800) ==> c(1280x 800+   0+   0) ==> out(1280x 800)

      scaler@  104000: disable

      snr@  106400: disable

      sharp@  106000: disable

      cdc@  108000: disable

      gtm disable
      vch0_cdc_icsc@      0: disable
      vch0_cdc_icsc@      0: disable

      dlc@  115000: disable
      gamma0@  116000:
            gamma_tbl off

      fcm@  111000: disable csc: disable
      vch0_fcm_icsc@      0: disable
      vch0_fcm_ocsc@      0: disable

      vch0_csc@  100800: enable
            in:  colorfmt: rgb, eotf: bt709, colorspace: BT709, range: full
            out: colorfmt: rgb, eotf: bt709, colorspace: BT709, range: full
plane[101]: plane-1-vch1(0) disable
      crtc=(null)
      rotation=1
      normalized-zpos=0
```

```
    ovl@ 121000: disable

    tfbd@ 125400: disable

    scaler@ 124000: disable

    vch1_csc@ 120800: disable
plane[107]: plane-2-uch0(0) disable
    crtc=(null)
    rotation=1
    normalized-zpos=0

    ovl@ 1c1000: disable

    tfbd@ 1c5400: disable

    scaler@ 1c4000: disable

    cdc@ 1c8000: disable

    uch0_csc@ 1c0800: disable
plane[113]: plane-3-uch2(0) disable
    crtc=(null)
    rotation=1
    normalized-zpos=0

    ovl@ 201000: disable

    scaler@ 204000: disable

    uch2_csc@ 200800: disable
blender@ 281000: enable
     pipe_id | enable | channel | premult |      crtc-pos
    ---------+--------+---------+---------+----------------------
        0  |   true |    0    |  false  | 1280x 800+   0+   0



    deband@ 287000: disable
    smbl0@ 28b000:
        smbl_tbl off, dimming: 256, backlight: 0
    gamma0@ 289000:
        gamma_tbl off
    disp0_gamma_csc@     0: enable
        in:  colorfmt: rgb, eotf: bt709, colorspace: BT709, range: full
        out: colorfmt: rgb, eotf: bt709, colorspace: BT709, range: full
        bcsh: 50 50 50 50
    dither@ 288000: disable
    fmt@ 285000: disable

plane[120]: plane-0-vch2(1) disable
    crtc=(null)
    rotation=1
    normalized-zpos=0

    ovl@ 141000: disable

    scaler@ 144000: disable

    snr@ 146400: disable
```

```
    fcm@ 151000: disable csc: disable
    vch2_fcm_icsc@    0: disable
    vch2_fcm_ocsc@    0: disable

    vch2_csc@ 140800: disable
plane[127]: plane-1-uch1(1) disable
    crtc=(null)
    rotation=1
    normalized-zpos=0

    ovl@ 1e1000: disable

    tfbd@ 1e5400: disable

    scaler@ 1e4000: disable

    uch1_csc@ 1e0800: disable
plane[133]: plane-2-uch3(1) disable
    crtc=(null)
    rotation=1
    normalized-zpos=0

    ovl@ 221000: disable

    scaler@ 224000: disable

    uch3_csc@ 220800: disable
blender@ 2a1000: disable
    smbl1@ 2ab000:
        smbl_tbl off, dimming: 256, backlight: 0
    gamma1@ 2a9000:
        gamma_tbl off
    disp1_gamma_csc@    0: disable
    dither@ 2a8000: disable
    fmt@ 2a5000: disable

crtc[100]: DE-0
    on: 1280x800@30&600Mhz->tcon0 irqcnt=3848 err=0
      format_space: 0 yuv_sampling: 0 eotf:1 cs: 1 color_range: 1 data_bits: 0
      vsync: 3848 last rcq at vsync: 3373
    wb off
crtc[126]: DE-1
    off
```

参数解析：

plane：

1. 图层参数信息

```
        layer_id | fb_id |     src-pos     |     crtc-pos     | blend_mode |  alpha |   color
        ---------+-------+-----------------+------------------+------------+--------+----------
          0   |  164  | 1280x 800+   0+   0 | 1280x 800+   0+   0 |   premult |   255  | 0x00000000
```

主要包括 src 原图的 crop 信息，和 crtc 显示区域。

2. plane 模块信息

```
scaler@ 104000: disable
xxx@ 偏移: disable/enable
```

这里表示未开启 plane 的 scaler 缩放功能

3. 颜色空间信息

```
vch0_csc@ 100800: enable
        in:  colorfmt: rgb, eotf: bt709, colorspace: BT709, range: full
        out: colorfmt: rgb, eotf: bt709, colorspace: BT709, range: full
```

输入和输出对应的颜色空间格式转换

blender：

1. pipe line 信息

```
 pipe_id | enable | channel | premult |     crtc-pos
 ---------+--------+---------+---------+--------------------
    0  |  true |   0   | false | 1280x 800+  0+  0
```

一个 plane 最终会对应到一个 pipe，这里表示所有 pipe 最终混合的信息

2. blender/disp_out 模块信息

```
disp0_gamma_csc@     0: enable
        in:  colorfmt: rgb, eotf: bt709, colorspace: BT709, range: full
        out: colorfmt: rgb, eotf: bt709, colorspace: BT709, range: full
        bcsh: 50 50 50 50
```

模块使能和配置信息，bcsh，表示亮度、对比度、饱和度、色度信息

crtc：

1. 硬件相关信息

```
on: 1280x800@30&600Mhz->tcon0 irqcnt=3848 err=0
  format_space: 0 yuv_sampling: 0 eotf:1 cs: 1 color_range: 1 data_bits: 0
  vsync: 3848 last rcq at vsync: 3373
wb off
```

- 1280x800@30：设备输出分辨率为 1280x800 30 hz
- err：当前状态的缺数数量，硬件内部的检测标志
- format_space、yuv_sampling、eotf、cs、color_range、data_bits：输出设备格式信息
- vsync、irqcnt：vsync 计数

5.3 drm obj 信息

```
mount -t debugfs none /sys/kernel/debug
cat /sys/kernel/debug/dri/*/*
```

drm 原生提供的一些 obj（crtc、connecter、framebuffer 等信息）。

6 FAQ

Q: libdrm 如何获取?

A: https://dri.freedesktop.org/libdrm/

Q: modetest 如何获取

A: 在 libdrm 下 tests 目录下

Q: sunxi drm 与 gpu 有什么关系?

A: 本文档中提到的 sunxi drm 与全志 soc 的 gpu 驱动没有太多关系，是对 AW 显示硬件 (de/tcon/hdmi/dsi/edp 等) 的控制驱动，soc 内的 gpu 驱动部分版本也通过 drm 实现，但与本文档无关。

Q: 如何查看 DRM 驱动当前的状态?

A:

1. 使能 debugfs 后，可以在 cat /sys/kernel/debug/dri//下查看各个 drm 组件当前的状态。
2. 直接运行 modetest，会显示 drm 组件的状态及全部 property，可以了解对应硬件的状态。
3. cat /sys/class/drm/card0/device/crtc/status 可以查看 de 的状态
4. /sys/class/drm/card0/card0-XXXX-X 包含 connector 的节点，可以查看或操作 connector 硬件。

Q: 无显示，如何检查驱动配置是否正确?

A:

1. 可以检查内核启动 log，如果驱动正常初始化，会有以下打印

```
[    6.086460] [drm] Initialized sunxi-drm 3.0.0 20230901 for soc@3000000:sunxi-drm on minor 0
[    6.095919] [drm] sunxi_drm_bind ok
```

2. 如果没有正常初始化的 log，可以往前查看是否有相关报错 log，一般都是某些依赖设备的驱动未配置导致，如 iommu，tcon，tcon_top（vo），pwm，backlight，panel，regulator，combo phy 等。
3. 检查/sys/class/drm/下是否有对应的设备，一般 card0 就是 sunxi-drm，以及 card0-HDMI-A-X/card0-LVDS-X 等输出接口的 connector 是否注册成功。

著作权声明

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





（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。