Application Note

在 Jacinot6 SOC 上集成 VISION SDK 和 PSDK 之间的虚拟 DRM

Fredy Zhang, Joe Shen, and Peter Li

摘要

直接渲染管理器 (DRM) 广泛应用于用户空间图形堆栈。在不允许 A15 访问 DSS 的设置中，无法使用正常工作的 omapdrm 驱动程序来显示内容。虚拟 DRM 是一个驱动程序框架，用于在此类设置中向 Linux® 用户空间显示 DRM 器件，从而使 Linux DRM 应用程序能够继续运行。

本文中所讨论的工程配套资料和源代码可从以下 URL 下载：https://www.ti.com/cn/lit/zip/spracx5。

内容

商标

1 简介

DRM/KMS 框架专用于管理显示、图形和合成子系统，如图 1-1 所示。

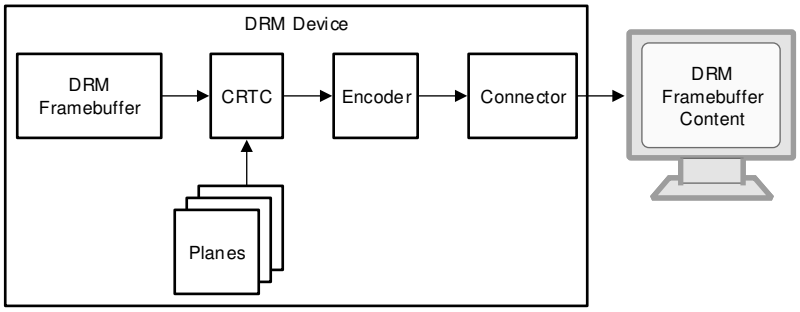

图 1-1. DRM/KMS 架构

借助其他 Linux 多媒体框架和应用程序，通常可使用 DRM/KMS 框架执行以下操作：

- 利用硬件加速功能合成动画内容。
- 控制显示界面和外部显示器，包括其设置（分辨率、频率、多屏幕等）。
- 在显示面板或 HDMI 输出上显示此动画内容。

DRM 器件：负责聚合其他组件。向用户空间显示的器件（处理所有用户空间请求。）

DRM 帧缓冲区：此为标准对象，存储有关要显示的内容的信息。

CRTC：CRTC 表示 CRT 控制器，它将帧缓冲区内容扫描至一个或多个显示器，并更新帧缓冲区。

层面：一个层面就是一个图像层。

编码器：负责将帧转换为适当的格式，以便通过连接器传输。

连接器：表示显示器连接器（HDMI、DP、VGA、DVI 等），将信号传输至显示器。检测显示器的连接/拆卸。显示显示器支持的模式。

在 vision SDK Linux 中，DSS 由在 IPU 上运行的软件控制。因此，需要禁用 omapdrm，并且基于 Linux 的 DRM 应用程序会停止正常运行，因为没有能够进行模式设置（显示内容）的 DRM 器件。引入了一个虚拟 DRM 框架来创建多个能够进行模式设置的 DRM 器件，并将它们显示在用户空间中。

使用 vDRM 框架，一方面，vDRM 支持 Linux 显示。另一方面，M4 可以控制 DSS 硬件。因此，当 M4 启动时，它可以按 M4 显示内容。

表 1-1 显示 PSDKLA 和 VISION SDK 的 DRM 比较。

表 1-1. PSDKLA 与 VISION SDK 的 DRM 比较

类型	PSDKLA	VISION SDK
DRM	DRM	虚拟 DRM
DSS	由 A15 控制 (Linux)	由 M4 控制 (RTOS)
Omapdrm 支持	是	否
Fb0	是	否

1.1 标准 DRM 框架

通常，IVI 系统可能包括三个部分：HMI 层面、标识层面和 RVC 视频层面。仪表组系统可能包括两个部分：HMI 层面和动画层面。

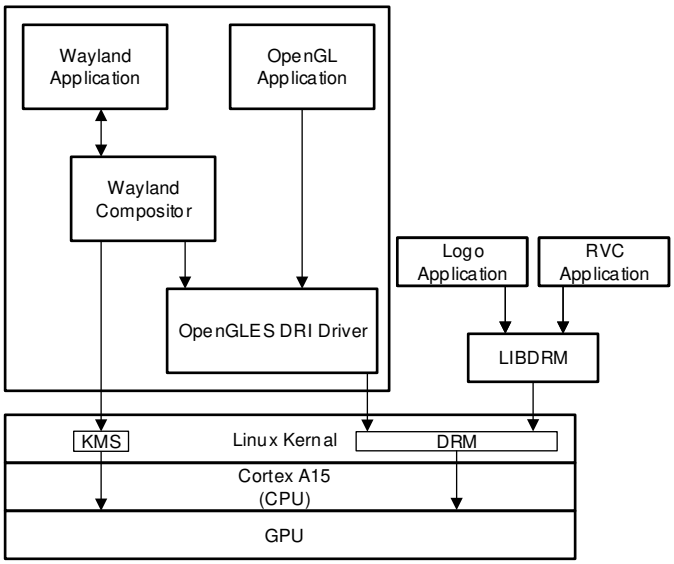

图 1-2. 标准框架

如图 1-2 所示，标准框架（即直接渲染管理器 (DRM)）驻留在内核空间，因此，用户空间程序必须使用内核系统调用来请求其服务。创建了一个名为 *libdrm* 的库，以方便用户空间程序与 DRM 子系统进行交互。该库只是一个包装器，它为 DRM API 的每个 ioctl 提供一个用 C 编写的函数，并提供常量、结构和其他辅助元素。

1.2 基于 vDRM 的框架

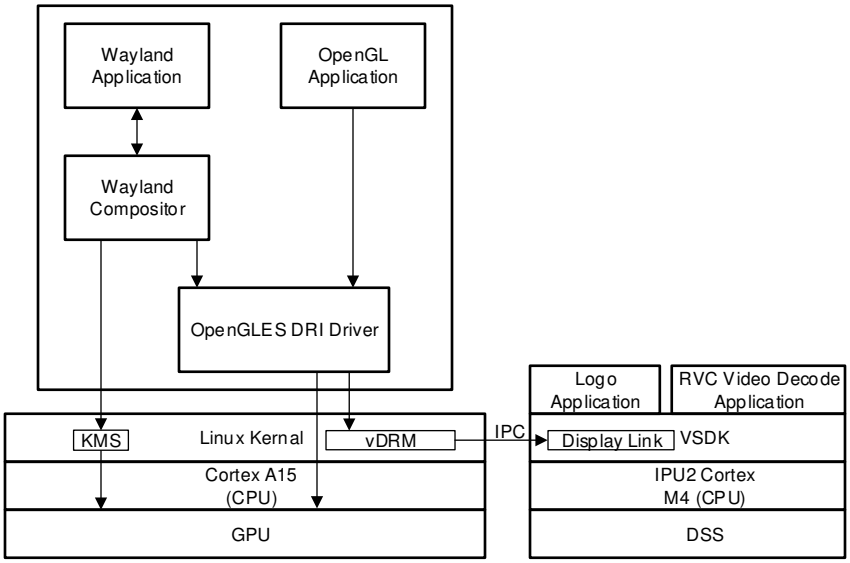

图 1-3. vDRM 框架

在此框架中，需要在 Linux 中禁用 omapdrm，因为 DSS 由在 IPU 上运行的软件控制。基于 DRM 的 Linux 应用程序将无法工作，因为没有能够进行模式设置（显示内容）的 DRM 器件。

虚拟 DRM 可创建多个能够进行模式设置的 DRM 器件，并将它们显示在用户空间中。每个 DRM 器件可包含多个 DRM 连接器，每个连接器能够配置为将预定义的分辨率和帧速率公开。每个 DRM 连接器都会在内部创建一个 DRM 编码器、一个 DRM 层面（主要）和一个 DRM CRTC，这些是 DRM API 正常工作所必需的。

此外，每个 DRM 器件创建一个 vdrm 控制器器件，Linux 应用程序可打开该器件以读取 DRM 应用程序提交的缓冲区。Vision SDK 可使用 dispDistSrcLink 的多个实例运行链（用例），其中每个链接读取一个 vdrm 控制器器件，以获取 DRM 应用程序提交给虚拟 DRM 器件中特定 CRTC 的缓冲区。

Linux 应用程序可以继续调用 DRM API 以在 DRM CRTC 上显示 DRM 帧缓冲区，即使 vision SDK 应用程序/链未运行，或者运行的链不包含与 CRTC 关联的 dispDistSrcLink。

从 VISION SDK 0304 开始，SDK 中提供了 vDRM 框架支持。

2 在 Linux 上基于 vDRM 显示内容

若要使 Linux 显示内容，需要在 M4 上运行 dispDistSrcLink 和 DisplayLink，以便 Linux 应用程序缓冲区可以传输至 M4 进行显示。使用以下步骤运行 Linux 应用程序：

1. 在 Linux 终端上运行 apps.out。
2. # 按 1 以选择单摄像头用例。
3. # 按 8 以选择 dispdist 用例。
4. 按 ctrl+z 以返回 Linux 终端。
5. 运行 Linux 应用程序。

SDK 还提供了快速启动方法来运行用例。使用以下步骤运行此用例。

1. 更改文件系统脚本：init-demo.sh (/home/root)：
   a. 从 LAUNCH_DEMO=0 更改为 LAUNCH_DEMO=1。
2. 重新启动主板后，系统便可自动运行 apps.out。

3 Linux 上基于 vDRM 的多媒体支持

由于 IVI/cluster/ADAS，项目对启动时间有特定的要求。与标准 DRM 显示框架不同。vDRM 框架可以调整多核架构并改善启动时性能。

对于 PSDKLA + VISION-SDK 架构，我们通常使用早期启动晚期附加的模式。图 3-1 显示了启动流程。

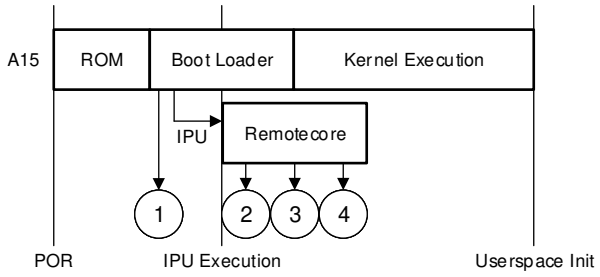

图 3-1. 启动流程

某些基于 omapdrm 的应用程序不能与虚拟 DRM 一同使用。因此，我们需要调整 vDRM 要求。以下是一些示例：Gstreamer/viddec3test/modetest/kmscube。可以从附件下载这些二进制文件。

3.1 Gstreamer

对于 gstreamer，它不能与预构建的 gstreamer drm 分配器一起工作。它需要 Gstreamer OmapDRM 分配器支持。我们提供了一个补丁来支持此功能。

使用以下步骤运行解码功能：

1. 在 VISON SDK 中启用 IPUMM。

   PROCESSOR_SDK_VISION_03_05_00_00/vision_sdk/apps/configs/tda2xx_evm_linux_all/cfg.mk

```
# Both IVAHD_INCLUDE & IPUMM_INCLUDE should not be set to "yes"
# Only one should be enabled to avoid IVA-HD resource conflict
IPUMM_INCLUDE=yes
IVAHD_INCLUDE=no
```

2. 启动主板，运行如下所示的命令：

```
root@dra7xx-evm# cd /opt/vision_sdk
root@dra7xx-evm# ./vision_sdk_load.sh
root@dra7xx-evm# ./apps.out
# press 1 to select : single camera usecases
# press 8 to select :  dispDistSrc -> display usecase
root@dra7xx-evm#  gst-launch-1.0 playbin uri=file:///home/root/test.mp4 video-sink=waylandsink
```

3.2 viddec3test

如果构建过程中包含 IPUMM，则可在 ssh 终端上运行以下任何一个命令来验证多媒体。

```
root@dra7xx-evm# viddec3test -w 640x480 --fps 24 /usr/share/ti/video/TearOfSteel-Short-1920x800.mov
```

3.3 modetest

- libdrm 库提供的工具 **modetest** 用于：
  - 列出所有显示功能：CRTC、编码器和连接器（DSI、DPI、HDMI 等）、层面、模式等
  - 执行基本测试：显示测试模式、显示 2 层、执行 vsync 测试
  - 指定视频模式：分辨率和刷新率

对于 vDRM 框架，您可以运行以下操作以查看每个卡的 DRM 相关信息：

```
root@dra7xx-evm# modetest -n /dev/dri/card1
root@dra7xx-evm# modetest -n /dev/dri/card1
root@dra7xx-evm# modetest -n /dev/dri/card2
```

3.4 kmscube

kmscube 是一个小型演示程序，介绍了如何在没有像 X11、wayland 或类似合成器的情况下，使用 DRM/KMS（内核模式设置）、GBM（图形缓冲区管理器）和 EGL 来驱动裸机图形，以便使用 OpenGL 或 OpenGL ES 呈现内容。

对于 vDRM 框架，请使用以下命令运行 kmscube。如果要运行 kmscube，请确保 weston 已停止（默认情况下，weston 在 /dev/dri/card0 上运行）。

```
root@dra7xx-evm#  kmscube -d /dev/dri/card0
```

4 显示基于 Weston 的应用

默认情况下，Weston 在 /dev/dri/card0 上运行。用户使用 vDRM 显示用户应用的示例。

请检查文件：/etc/powervr.ini。

```
[default]
#WindowSystem=libpvrws_WAYLAND.so

[weston]
DbmDriverName=vdrm

[kmscube]
DbmDriverName=vdrm
GbmNumBuffers=5
```

确保在此文件中配置 Weston。然后，您可以检查 Weston 状态并运行 Weston 应用程序。

5 显示基于 EGL 的应用程序

配置基于 EGL 的应用程序，请参阅文件：/etc/powervr.ini。

```
[default]
#WindowSystem=libpvrws_WAYLAND.so

[weston]
DbmDriverName=vdrm

[kmscube]
DbmDriverName=vdrm
GbmNumBuffers=5
```

如果要添加用户应用，请添加应用配置，如下所示：

```
[user_app]
DbmDriverName=vdrm
GbmNumBuffers=5
```

除了配置之外，还需要在应用中配置器件名称。提供了一个补丁 (0001-add-flag-for-device-node-name.patch) 以供您参考。

6 跨 PSDKLA 和 VISION-SDK 的交互式显示

对于那些通常使用不同管道通道的用例，您希望显示 Linux 和 VISION-SDK 应用程序。您可以在这些用例中更改 Alpha 和 Zorder 设置。

6.1 ALPHA 设置

Address Offset	0x0000 0074		
Physical Address	0x5800 1074	**Instance**	DISPC
Description	The register defines the global alpha value for the graphics and three video pipelines. Shadow register, updated on VFP start period of primary LCD or VFP start period of the third LCD or VFP start period of the secondary LCD or VFP start period of the third LCD or EVSYNC or when DISPC_CONTROL2.GOWB is set to 1 by software and current WB frame is finished (no more data in the write-back pipeline). The synchronization event is defined based on the output using the pipeline: primary LCD, secondary LCD, third LCD, TV output or write-back to the memory for each bit field depending on the association of the each pipeline with the primary LCD, secondary LCD or TV output.		
Type	RW		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
VID3GLOBALALPHA	VID2GLOBALALPHA	VID1GLOBALALPHA	GFXGLOBALALPHA

Bits	Field Name	Description	Type	Reset
31:24	VID3GLOBALALPHA	Global alpha value from 0 to 255. 0 corresponds to fully transparent and 255 to fully opaque.	RW	0xFF
23:16	VID2GLOBALALPHA	Global alpha value from 0 to 255. 0 corresponds to fully transparent and 255 to fully opaque.	RW	0xFF
15:8	VID1GLOBALALPHA	Global alpha value from 0 to 255. 0 corresponds to fully transparent and 255 to fully opaque.	RW	0xFF
7:0	GFXGLOBALALPHA	Global alpha value from 0 to 255. 0 corresponds to fully transparent and 255 to fully opaque.	RW	0xFF

图 6-1. Alpha 设置

6.2 ZORDER 设置：DISPC_xxx_ATTRIBUTES[26-27]

27:26	ZORDER	Z-Order defining the priority of the layer compared to others when overlaying. It is software responsibility to ensure that each layer connected to the same overlay manager has a different z-order value. If bit 25 is set to 0, the ZORDER bit field is ignored and replaced by the value 0. 0x0: Z-order 0: layer above solid background color and below layer with higher Z-order values. 0x1: Z-order 1: layer above layer with z-order value of 0 and below layers with z-order values of 2 and 3 0x3: Z-order 3: layer above all the other layers 0x2: Z-order 2: layer above layers with z-order value of 0 and 1 and below layer with z-order value of 3	RW	0x0

图 6-2. ZORDER 设置：DISPC_xxx_ATTRIBUTES[26-27]

7 双显示器演示

双显示器演示需要两个屏幕来显示不同的内容：一个屏幕用于仪表组，另一个屏幕用于环视。

附件提供了一个双显示器演示用例。

8 构建 Linux Vision SDK 文件系统

从 Vision-SDK 3.4 开始，文件系统构建过程发生了一些变动，以支持更小的文件系统。尽管作为 Processor-SDK Linux Automotive 的一部分提供的文件系统约为 700MB，但作为 Vision SDK 发行版一部分的文件系统约为 60MB。移除传统 ADAS 用例不需要的组件可减小空间大小。

按照以下说明重新构建 Vision SDK 文件系统：

1. 按照维基网站文章《Processor SDK Linux Automotive 软件开发人员指南》中的说明构建 Yocto 文件系统。
2. 将 Vision-SDK 的 linux-kernel-addon/fs-patches/yocto/meta-glsdk 文件夹中的补丁应用到 yocto 存储库中的 tisdk/sources/meta-glsdk 文件夹。
3. 将 Vision-SDK 的 linux-kernel-addon/fs-patches/yocto/meta-arago 文件夹中的补丁应用到 yocto 存储库中的 tisdk/sources/meta-arago 文件夹。
4. 按照维基网站文章《Processor SDK Linux Automotive 软件开发人员指南》中的说明，通过运行 bitbake 命令重新构建文件系统。
5. meta-arago 中的更改用于减小文件系统大小，而 meta-glsdk 中的更改用于基于 VDRM 和 VDRM+ IPUMM 的解码。

9 参考文献

- 《Processor SDK Linux Automotive 软件开发人员指南》

重要声明和免责声明

TI 提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保或其他要求。这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款 (https:www.ti.com/legal/termsofsale.html) 或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265