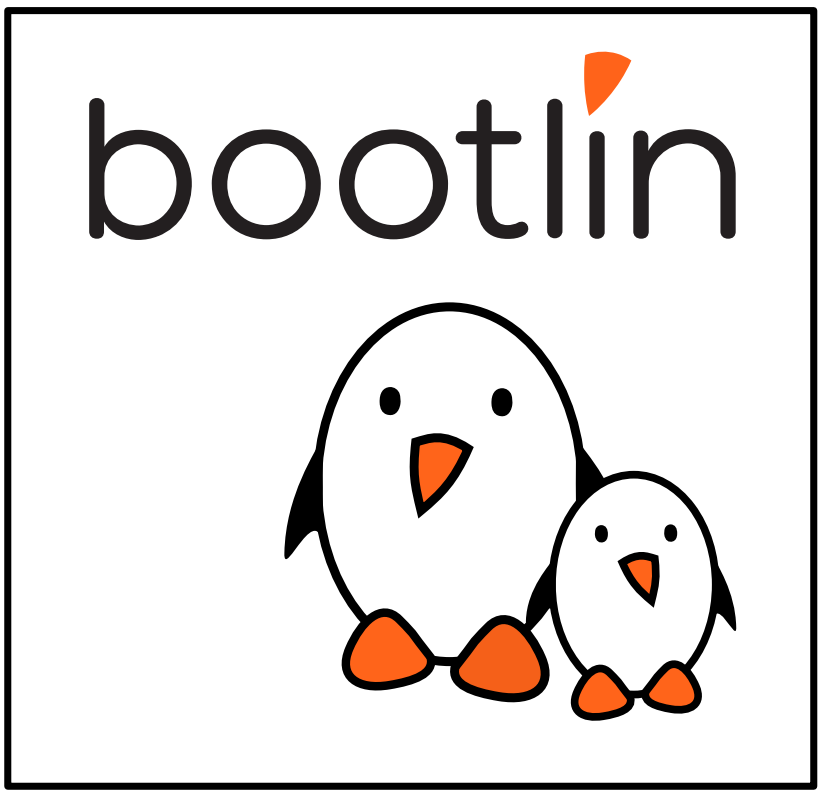
Graphic Testing without hardware: discovering the power of VKMS!



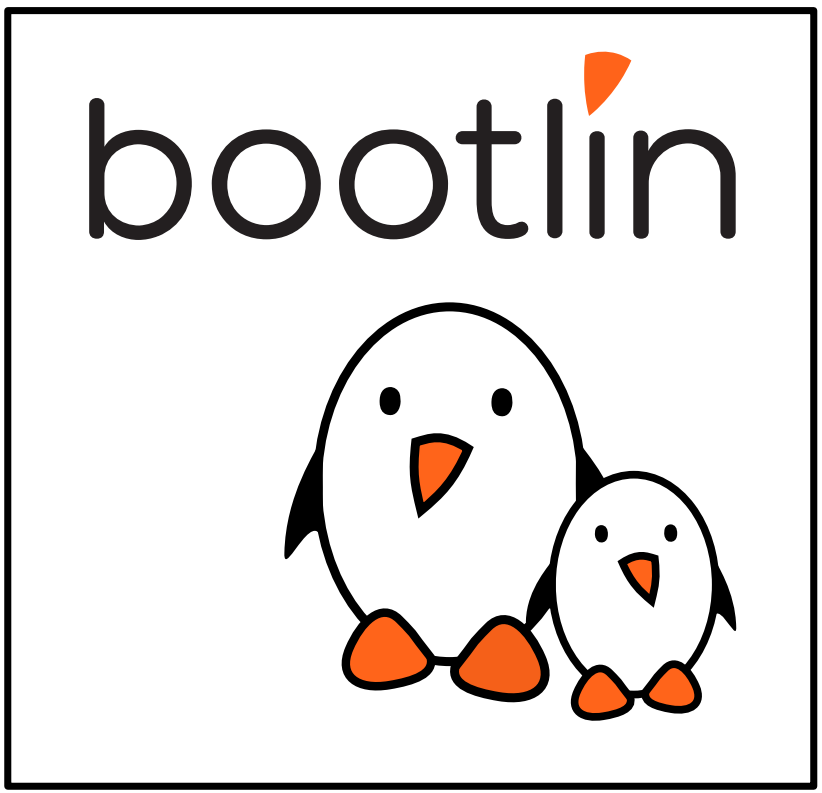
Louis Chauvet
louis.chauvet@bootlin.com

Louis Chauvet

- Embedded Linux engineer at Bootlin
  - Embedded Linux expertise
  - Zephyr expertise
  - Development, consulting and training
  - Strong open-source focus
- VKMS Maintainer
- Living in **Toulouse**

Display testing

- What do we want to test?
- How to do it?



What do we want to test?

- Anything with a screen
- App which don't know exactly what is the hardware it will be ran on

How to do it?

Without VKMS	With VKMS
1. Buy boards 2. Setup board 3. Debug 4. Go to step 1.	1. Run VM 2. Set VKMS config 3. Debug 4. Go to step 1.
Pros: Real hardware **Cons**: Expensive, physical manipulations, hard to integrate in CI	**Pros**: Free, easy to automate **Cons**: Not the real hardware

VKMS, what it is?

VKMS, what it is?

VKMS: **V**irtual **K**ernel **M**ode **S**etting

> VKMS is a software-only model of a KMS driver that is useful for testing and for running X (or similar) on headless machines. VKMS aims to enable a virtual display with no need of a hardware display capability, releasing the GPU in DRM API tests.
>
> — Documentation/gpu/vkms.rst

VKMS, what it is?

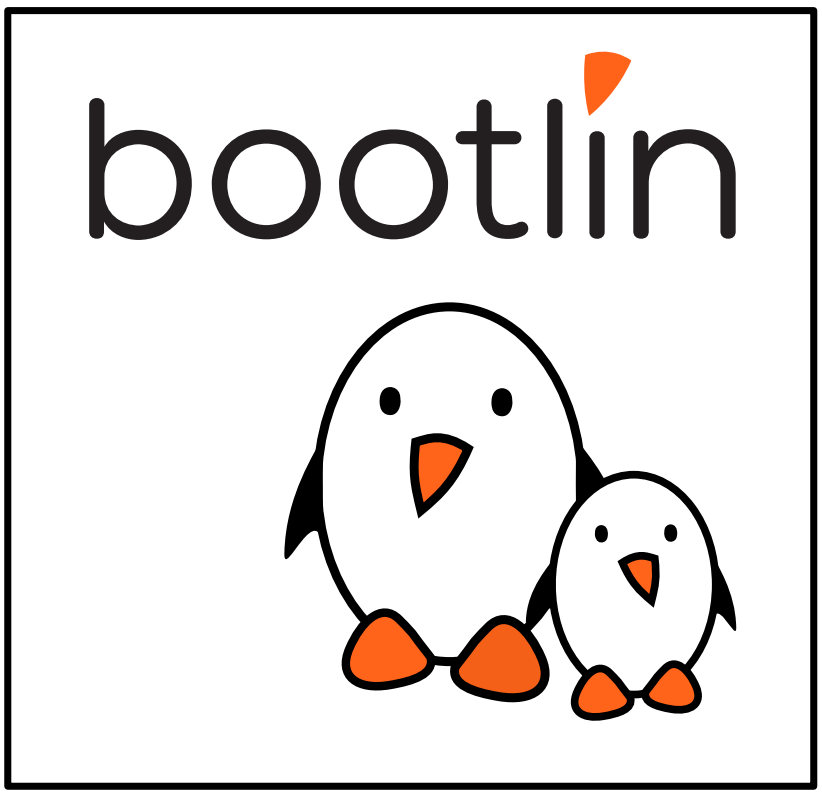
Virtual - No hardware required, can be used in virtual machine

Kernel Modesetting - Real kernel interface, no hack on your application

Main goal: Testing userspace without hardware

Current VKMS Capabilities

- Plane composition
- Color conversion
- Writeback
- Kernel arguments
- Minimal configuration
- Maximal configuration



Plane composition

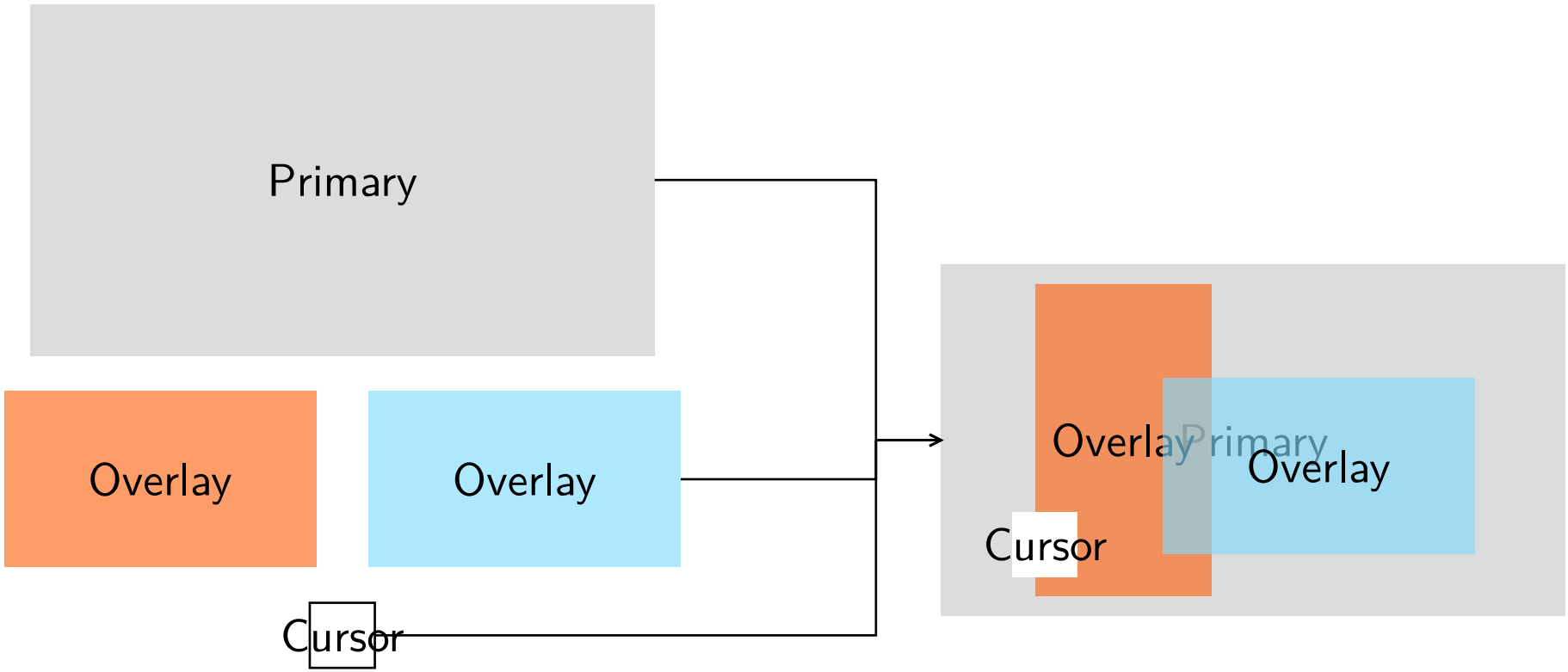


Figure 1: Plane composition

Color conversion

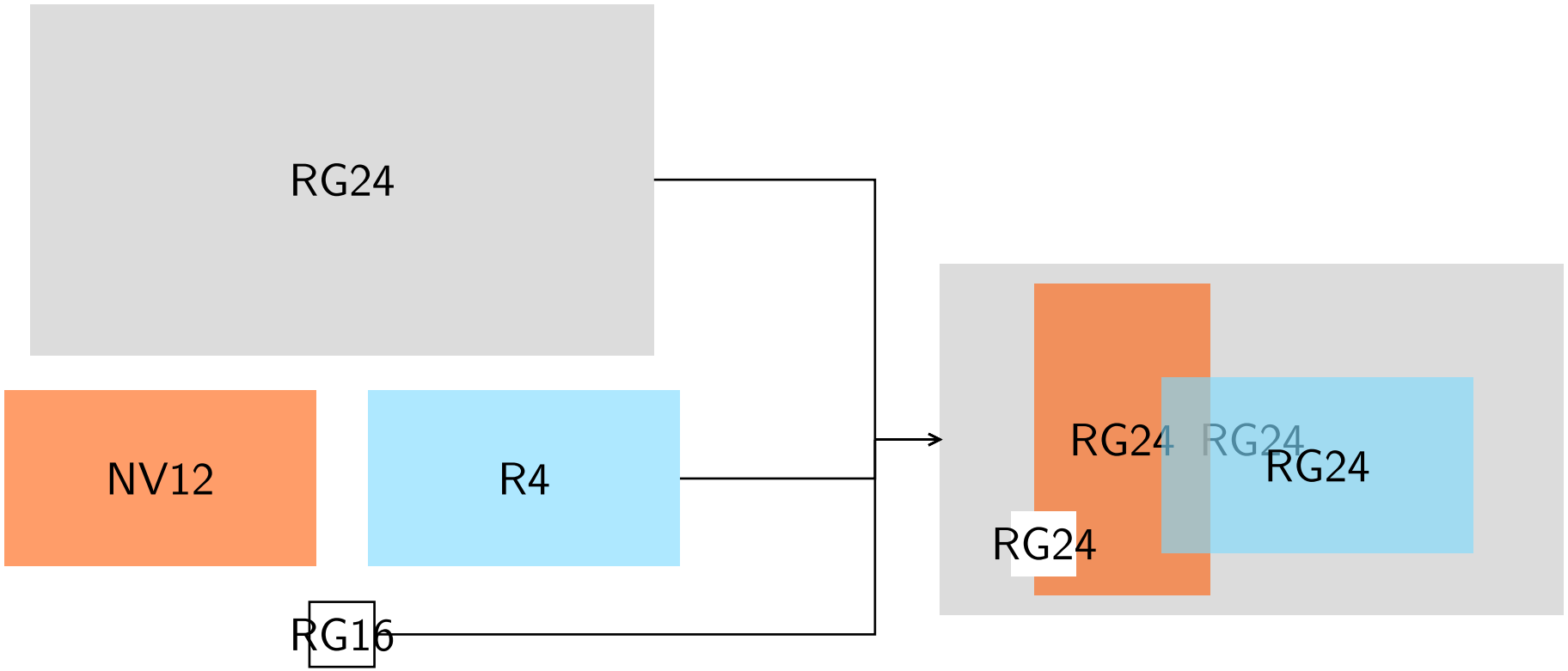


Figure 2: Plane composition

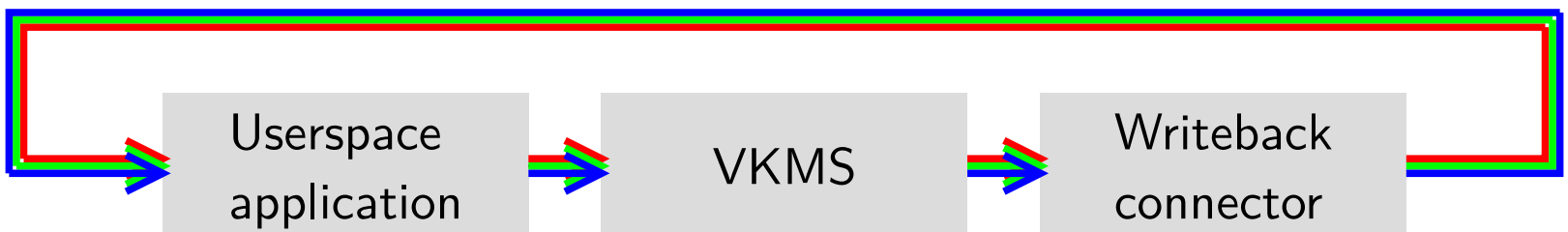


Figure 3: Writeback connector

Kernel arguments

- `enable_cursor` - `true`

  Add a cursor plane
- `enable_writeback` - `true`

  Add a writeback connector
- `enable_overlay` - `false`

  Add multiple planes

Minimal configuration

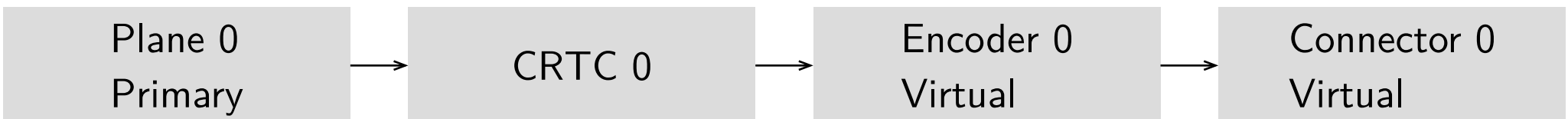


Figure 4: `enable_cursor=false`, `enable_writeback=false`, `enable_overlay=false`

Maximal configuration

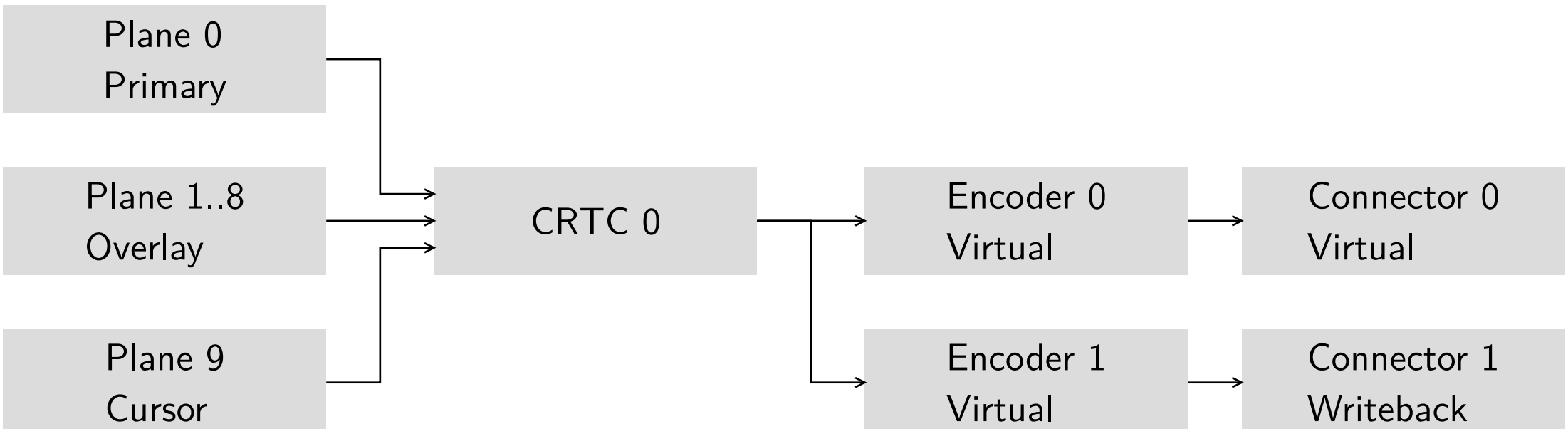
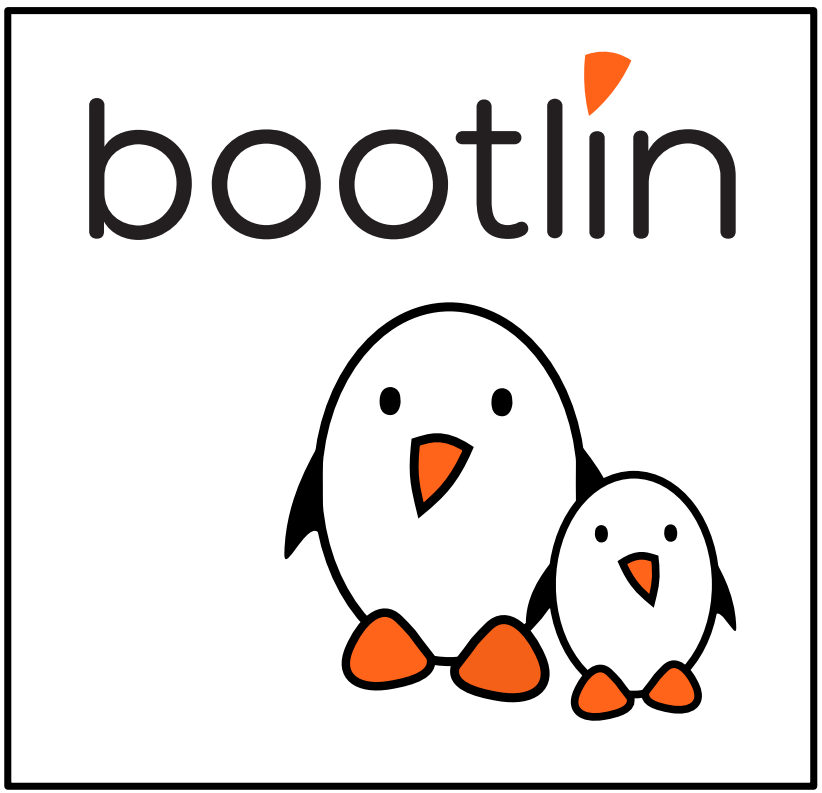


Figure 5: `enable_cursor=true`, `enable_writeback=true`, `enable_overlay=true`

Real life examples

- Texas Instrument - AM6232
- Framework 13 - intel



Real life examples - AM6232

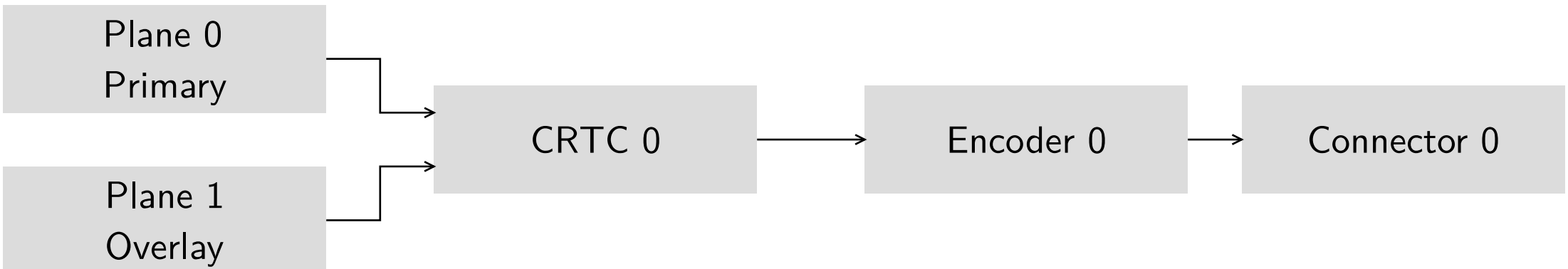


Figure 6: AM6232

Real life examples - Framework 13 - intel

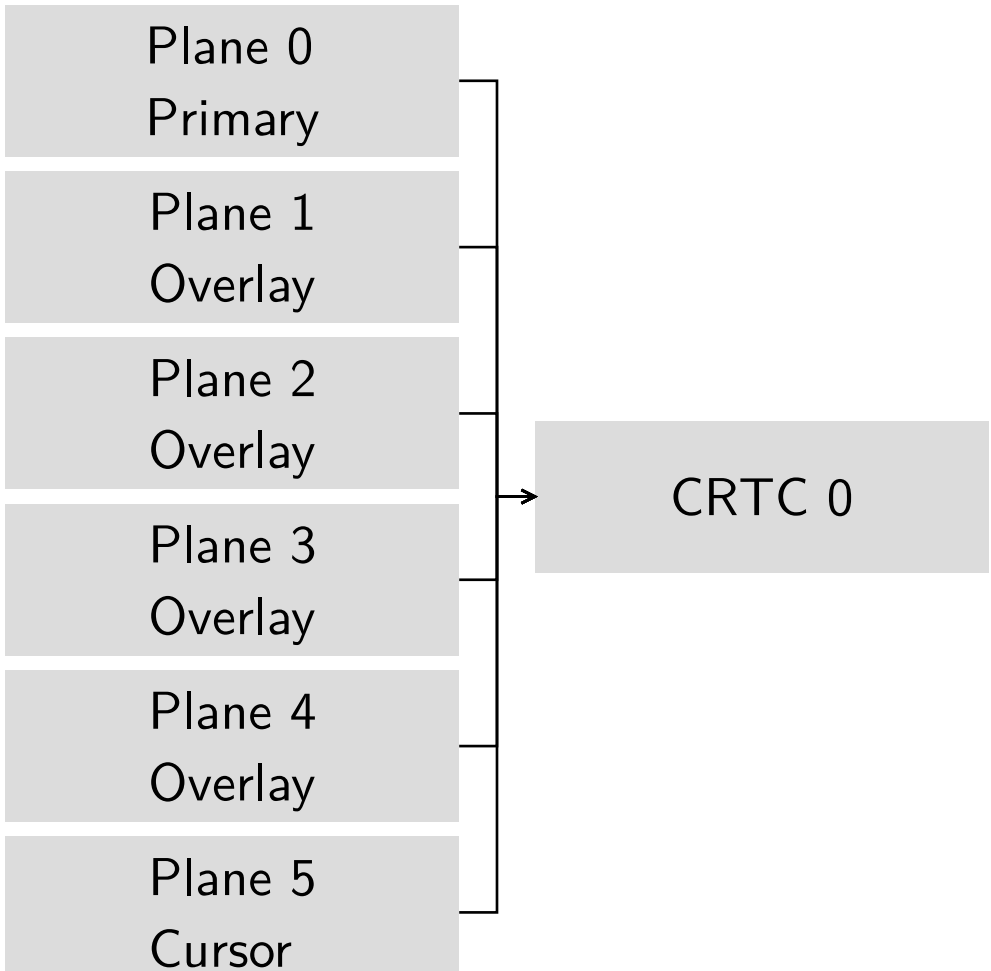


Figure 7: Framework 13

Real life examples - Framework 13 - intel

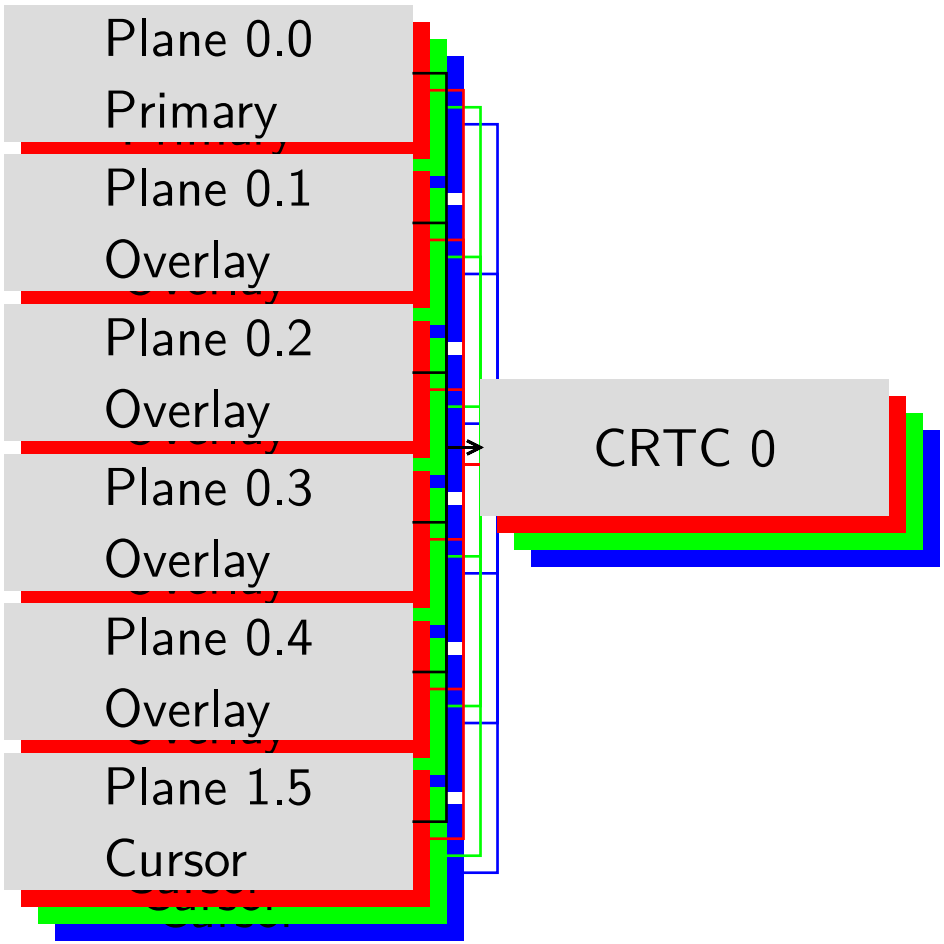


Figure 8: Framework 13

Real life examples - Framework 13 - intel

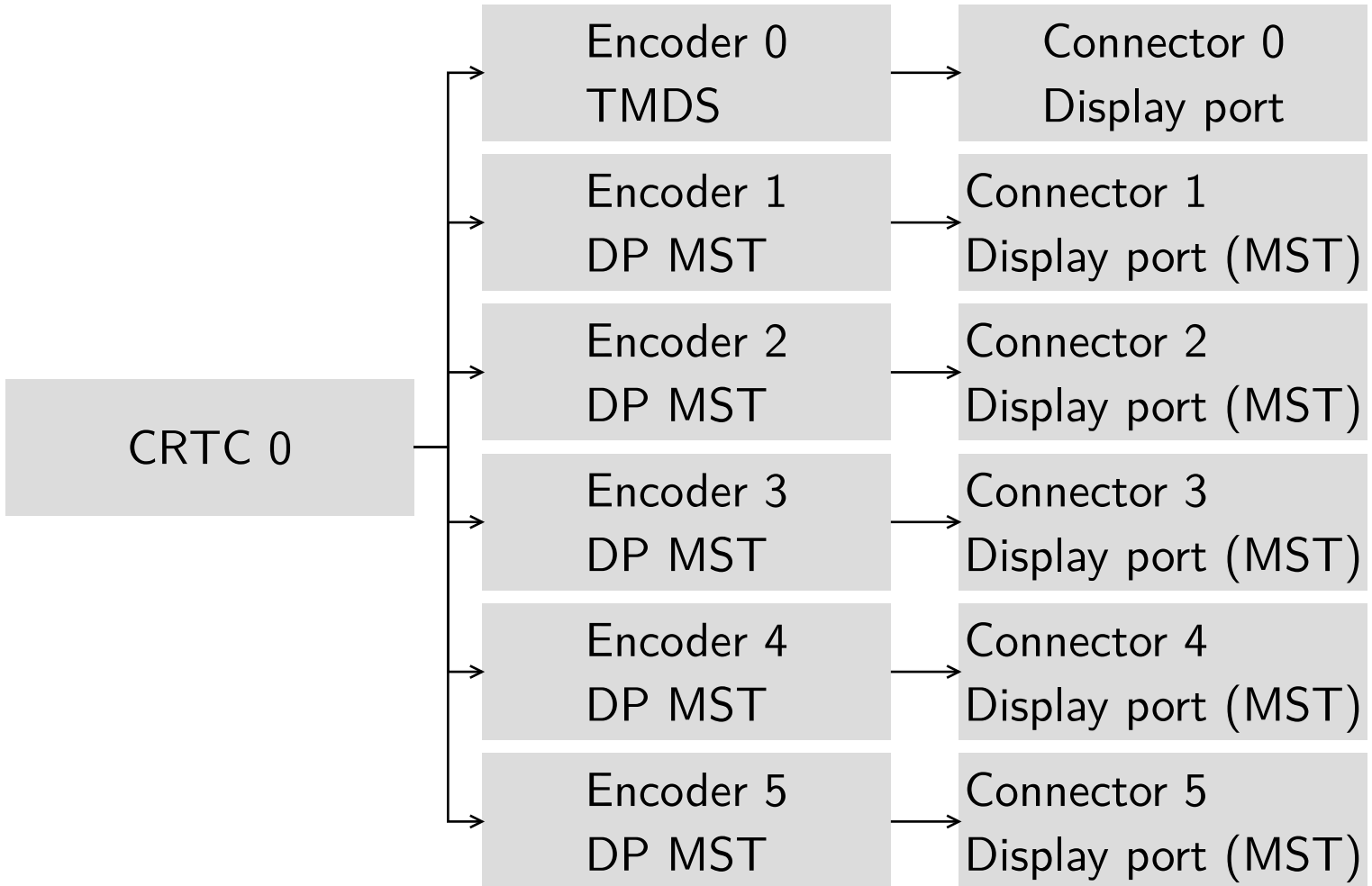


Figure 9: Framework 13

Real life examples - Framework 13 - intel

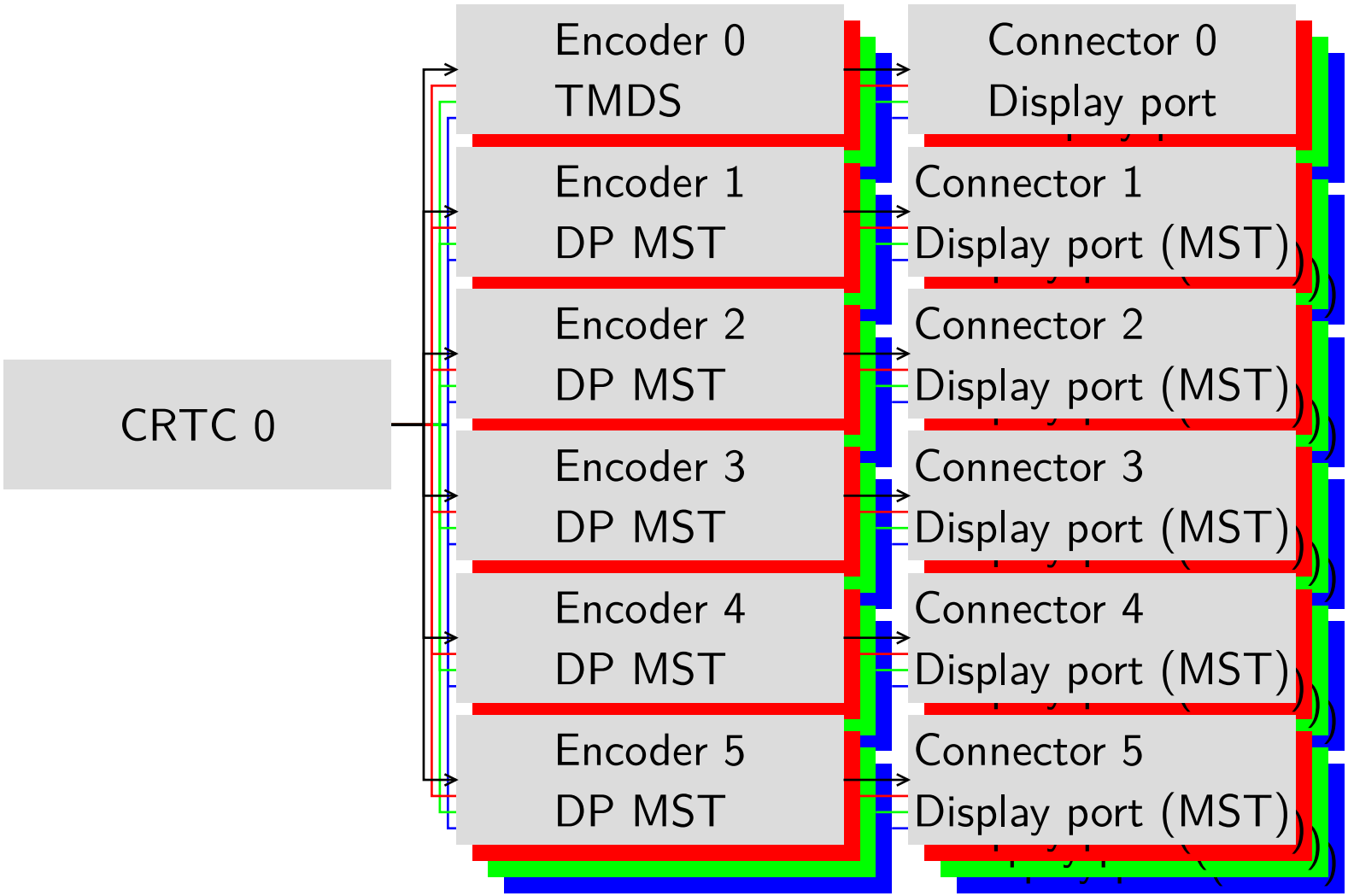


Figure 10: Framework 13

ConfigFS interface

- Upstreaming status

Upstreaming status

- First work by Rodrigo Siqueira, Sumera Priyadarsini, Marius Vlad, Jim Shargo (2019-2023)
- José Exposito and me continued the work after (2024-today)

Upstreaming status

- First work by Rodrigo Siqueira, Sumera Priyadarsini, Marius Vlad, Jim Shargo (2019-2023)
- José Exposito and me continued the work after (2024-today)

Last iteration: [PATCH v6 00/16] drm/vkms: Add configfs support

Repository: github.com/Fomys/linux/tree/review/20250901122541.9983-1-jose.exposito89@gmail.com

Demo time - probing

- New kernel argument: `create_default_dev`, disable or enable the creation of the default device to ensure backwards compatibility

```
$ modprobe vkms create_default_dev=0
```

Demo time - Creating a device

- Creating a device is as simple as creating a folder in ConfigFS:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV
```

Demo time - Creating a device

- Creating a device is as simple as creating a folder in ConfigFS:

```
$ mkdir /sys/kernel/config/vkms/MY_DEV
```

```
$ tree /sys/kernel/config/vkms/MY_DEV
/sys/kernel/config/vkms/DEV_1/
├── connectors/
├── crtcs/
├── enabled
├── encoders/
└── planes/
```

Demo time - Enabling a device

- Enabling a device is done using the enabled file:

Demo time - Enabling a device

- Enabling a device is done using the enabled file:

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
```

Demo time - Enabling a device

- Enabling a device is done using the enabled file:

```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
(NULL device *): [drm] The number of planes must be between 1 and 31
```

But currently we did not configure it yet, DRM is complaining about missing planes

Pending patches

A lot of work is ready to go

- New color formats for planes and writeback
- More configuration for planes, connector, encoder

Repository: https://github.com/Fomys/linux/tree/b4/vkms-dynamic-connectors

It is already used!

- KWin, Mutter, DRM CI (upstream VKMS)
- Android Hardware Composer - https://android.googlesource.com/platform/platform_testing/+/refs/heads/android16-s2-release/tests/display/hcct/
- IGT - https://lore.kernel.org/all/20250807074550.6543-3-jose.exposito89@gmail.com/
- Android CI - https://gitlab.freedesktop.org/search?search=vkms&nav_source=navbar&project_id=5&group_id=1332&search_code=true&repository_ref=main
- ConfigFS support planned in KWin, Mutter (compositor and headset), Weston, ...

It is already used!

- KWin, Mutter, DRM CI (upstream VKMS)
- Android Hardware Composer - https://android.googlesource.com/platform/platform_testing/+/refs/heads/android16-s2-release/tests/display/hcct/
- IGT - https://lore.kernel.org/all/20250807074550.6543-3-jose.exposito89@gmail.com/
- Android CI - https://gitlab.freedesktop.org/search?search=vkms&nav_source=navbar&project_id=5&group_id=1332&search_code=true&repository_ref=main
- ConfigFS support planned in KWin, Mutter (compositor and headset), Weston, ...

We need reviews!

Demo time


```
$ modprobe vkms create_default_dev=0
$ mkdir /sys/kernel/config/vkms/MY_DEV
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_1
$ mkdir /sys/kernel/config/vkms/MY_DEV/planes/PLA_2
$ mkdir /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1
$ mkdir /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1
$ mkdir /sys/kernel/config/vkms/MY_DEV/connectors/CON_1
$ mkdir /sys/kernel/config/vkms/MY_DEV/connectors/CON_2
```

Demo time - continuation


```
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/planes/PLA_1/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/planes/PLA_2/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/crtcs/CRTC_1/ /sys/kernel/config/vkms/
MY_DEV/encoders/ENC_1/possible_crtcs/
$ ln -s /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1/ /sys/kernel/config/vkms/
MY_DEV/connectors/CON_1/possible_encoders/
$ ln -s /sys/kernel/config/vkms/MY_DEV/encoders/ENC_1/ /sys/kernel/config/vkms/
MY_DEV/connectors/CON_2/possible_encoders/
```

Demo time - plane configuration


```
$ tree /sys/kernel/config/vkms/MY_DEV/planes/
/sys/kernel/config/vkms/MY_DEV/planes/
└── PLA_1
    ├── default_color_encoding      # default for color encoding
    ├── default_color_range         # default for color range
    ├── default_rotation            # default for rotation
    ├── possible_crtcs
    │   └── CRTC_1 -> ../../../../../vkms/MY_DEV/crtcs/CRTC_1
    ├── supported_color_encoding    # color encoding bitmask
    ├── supported_color_range       # color range bitmask
    ├── supported_formats           # +RG24 to enable RGB888, -RG24 to disable
    ├── supported_rotations         # rotation bitmask
    └── type                        # 0 = Overlay, 1 = Primary, 2 = Cursor
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/planes/PLA_1/type
$ echo 0 > /sys/kernel/config/vkms/MY_DEV/planes/PLA_2/type
```

Demo time - plane color format configuration

- Only support RGB888 for primary plane

```
$ echo '-*' > /sys/kernel/config/vkms/MY_DEV/planes/PLA_1/supported_formats
$ echo '+RG24' > /sys/kernel/config/vkms/MY_DEV/planes/PLA_1/supported_formats
```

- Support everything except RGB888 for overlay plane

```
$ echo '+*' > /sys/kernel/config/vkms/MY_DEV/planes/PLA_2/supported_formats
$ echo '-RG24' > /sys/kernel/config/vkms/MY_DEV/planes/PLA_2/supported_formats
```

Demo time - encoder configuration


```
$ tree /sys/kernel/config/vkms/MY_DEV/encoders/
/sys/kernel/config/vkms/MY_DEV/encoders/
└── ENC_1
    ├── possible_crtcs
    │   └── CRTC_1 -> ../../../../../vkms/MY_DEV/crtcs/CRTC_1
    └── type            # 5 = virtual, 3 = LVDS, ...
```

Demo time - connector configuration


```
$ tree /sys/kernel/config/vkms/MY_DEV/connectors/
/sys/kernel/config/vkms/MY_DEV/connectors/
└── CON_1
    ├── status                  # 1 = connected, 2 = disconnected, 2 = unknown
    ├── type                    # 15 = virtual, 10 = DP, ...
    ├── supported_colorspace    # for HDR support (only for type=eDP, DP or
HDMI)
    ├── edid                    # raw edid content
    ├── edid_enabled            # 1 = use the provided edid, 0 = use default
mode list
    ├── possible_encoders
    │   └── ENC_1 -> ../../../../../vkms/MY_DEV/encoders/ENC_1
    ├── dynamic                 # 1 = dynamic connector (MST), 0 = static
    └── enabled                 # if dynamic, create/destroy this connector on
the fly
```

Demo time - dynamic connector


```
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/connectors/CON_2/dynamic
$ echo 0 > /sys/kernel/config/vkms/MY_DEV/connectors/CON_2/enabled
$ echo 1 > /sys/kernel/config/vkms/MY_DEV/enabled
$ modetest -M vkms | grep connected
45  0 connected Virtual-1       0x0   34  44

$ echo 1 > /sys/kernel/config/vkms/MY_DEV/connectors/CON_2/enabled
$ modetest -M vkms | grep connected
45  0 connected Virtual-1       0x0   34  44
46  0 connected Virtual-2       0x0   34  44
```

Questions? Suggestions? Comments?

Louis Chauvet
louis.chauvet@bootlin.com

https://bootlin.com/pub/conferences/