Shifting Media app development into high gear

Using virtual drivers to speed up development

Helen Koike
helen.koike@collabora.com
Summary

- Classic V4L2 API → Vivid Driver
- Media API (extension) → Vimc Driver

- Vimc:
  - Submodules
  - Current state
  - Configfs API
  - Future work
Summary

• Classic V4L2 API → Vivid Driver
• Media API (extension) → Vmcc Driver

• Vmcc:
  • Submodules
  • Current state
  • Configfs API
  • Future work
User space

/dev/video*

Kernel space

Driver

Physical device
User space

Kernel space

Physical device

APP

Read/Write/MMAP

IOCTLS

/dev/video*

Driver

Video stream

General configs
* Img fmt
* Buffers
* Video std
  *
* Frame rate
  *

Standard Controls
* Contrast
  *
* Brightness
  *
* Gamma
  *

Custom Controls
* DRV DEF 1
  *
* DRV DEF 2
  *
* DRV DEF 3
  *
The Virtual Video Test Driver (VIVID)
The Virtual Video Test Driver (VIVID)

User space

Kernel space

Physical device

/dev/video*
Vivid driver

<table>
<thead>
<tr>
<th>Vivid Controls</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Pattern</td>
<td>75% Color Bar</td>
</tr>
<tr>
<td>OSD Text Mode</td>
<td>All</td>
</tr>
<tr>
<td>Horizontal Movement</td>
<td>No Movement</td>
</tr>
<tr>
<td>Vertical Movement</td>
<td>No Movement</td>
</tr>
<tr>
<td>Show Border</td>
<td></td>
</tr>
<tr>
<td>Show Square</td>
<td></td>
</tr>
<tr>
<td>Insert V4L Code in Image</td>
<td>No</td>
</tr>
<tr>
<td>Insert EAV Code in Image</td>
<td>No</td>
</tr>
<tr>
<td>Sensor Flipped Horizontally</td>
<td>No</td>
</tr>
<tr>
<td>Sensor Flipped Vertically</td>
<td>No</td>
</tr>
<tr>
<td>Standard Aspect Ratio</td>
<td>4:3</td>
</tr>
<tr>
<td>Target Width</td>
<td>1280</td>
</tr>
<tr>
<td>Target Height</td>
<td>720</td>
</tr>
<tr>
<td>Transfer Function</td>
<td>Default</td>
</tr>
<tr>
<td>Y/C Ordering</td>
<td>Default</td>
</tr>
<tr>
<td>Quantization</td>
<td></td>
</tr>
<tr>
<td>Limited HLG Range (10/210)</td>
<td>No</td>
</tr>
<tr>
<td>Apply Alpha To Red Only</td>
<td>No</td>
</tr>
<tr>
<td>Enable Capture Cropping</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Frame: 41 fps; 5:00 Scale Factors: 1x1 Scale: 40
Summary

- Classic V4L2 API → Vivid Driver
- Media API (extension) → Vimc Driver

- Vimc:
  - Submodules
  - Current state
  - Configfs API
  - Future work
Media API Motivation

- Embedded systems: great variety of devices
- Hardware complexity
- Similar configuration for different entities
Sensor

Composer

Scaler

Contrast

Sepia

Greyscale
User space

Kernel space

Physical device

Driver

Video stream

General configs

* Img fmt
* Buffers
* Video std
* Frame rate

Standard Controls

* **Contrast**
* Brightness
* Gamma

Custom Controls

* DRV DEF 1
* DRV DEF 2
* DRV DEF 3

APP

Read/Write/MMAP

IOCTLs

/dev/video*

---
What is part of the device?
What is the topology?

/dev/v4l-subdev0
/dev/v4l-subdev14
/dev/video0
/dev/v4l-subdev31
/dev/v4l-subdev98
/dev/v4l-subdev6
/dev/video42
/dev/v4l-subdev21
/dev/video2
Media API

- `/dev/video0`
  - R/W/MMAP
  - IOCTLs

- `/dev/v4l-subdev0`
  - IOCTLs
  - contrast (Sepia)

- `/dev/v4l-subdev1`
  - IOCTLs
  - contrast (Greyscale)

- `/dev/v4l-subdev2`
  - IOCTLs
  - Scaler
User space

/dev/media*

Kernel space

Driver

Physical device
The Virtual Media Controller Driver (VIMC)
The Virtual Media Controller Driver (VIMC)

User space
/dev/media*

Kernel space

Physical device

?
Summary

- Classic V4L2 API → Vivid Driver
- Media API (extension) → Vimc Driver

- Vimc:
  - Submodules
  - Current state
  - Configfs API
  - Future work
User space

Kernel space

VIMC Configfs API

/dev/media0

/dev/media1

create

create

create
Vimc: submodules

Implementation of subdevice drivers are modularized and doesn’t need to alter Vimc’s core code.
**APP Developer**

Emulate different topologies to test the app in several scenarios
Summary

- Classic V4L2 API → Vivid Driver
- Media API (extension) → Virc Driver

- Virc:
  - Submodules
  - Current state
  - Configfs API
  - Future work
Vimc Driver

- Proposed by Laurent Pinchart for Outreachy in 2015
- Merged in Kernel 4.12
- Moving slowly
Vimc Driver: current state

- Basic standard submodules:
  - vimc-capture.ko
  - vimc-sensor.ko
  - vimc-debayer.ko
  - vimc-scaler.ko
- Hard-coded topology (re-compilation required)
Vimc Driver
Userspace tools

- **v4l-utils**
  
  ```
  media-ctl -d /dev/media0 --print-dot > /tmp/out.dot && xdot /tmp/out.dot 
  media-ctl -d /dev/media0 -V "'Sensor A':0[fmt:RGB888_1X24/600x600]"
  media-ctl -v -d /dev/media0 --links "'Debayer A':1->'Scaler':0 [0]"
  ```

- **Yavta (Yet Another V4L2 Test Application)**
  
  ```
  yavta --format RGB24 --size 600x600 /dev/video0
  ```
Summary

- Classic V4L2 API → Vivid Driver
- Media API (extension) → Vimc Driver

Vimc:
- Submodules
- Current state
- Configfs API
- Future work
Configfs API: Topology

- **Entities**
  - Name
  - Submodule

- **Pads**
  - Source
  - Sink

- **Links**
mkdir "MEDIA_NAME"
mkdir "SUBMOD:NAME"

- 'Sensor A'
  - vimc-sensor.ko
  - 0

- 0
  - 'Debayer A'
    - vimc-debayer.ko
    - 1

- 0
  - 'Raw Capture 0'
    - vimc-capture.ko
    - 1
mkdir "NAME1:PAD->NAME2:PAD"
Summary

- Classic V4L2 API → Vivid Driver
- Media API (extension) → Vmcs Driver

Vimcs:
- Submodules
- Current state
- Configfs API

Future work
Vimc Driver: future work

- API in Configfs (WIP)
- vimc-input.ko (WIP)
- Optimizations of img processing in the pipeline (WIP)
- GUI User space tool
- More standard submodules
- Add more V4L2 mechanisms / controls / options
Thank you!

Helen Koike
helen.koike@collabora.com
This work is licensed under https://creativecommons.org/licenses/by-sa/4.0/