Introduction

The V4L2 drivers tend to be very complex due to the complexity of the hardware: most devices have multiple ICs, export multiple device nodes in /dev, and create also non-V4L2 devices such as DVB, ALSA, FB, I2C and input (IR) devices.

Especially the fact that V4L2 drivers have to setup supporting ICs to do audio/video muxing/encoding/decoding makes it more complex than most. Usually these ICs are connected to the main bridge driver through one or more I2C buses, but other buses can also be used. Such devices are called 'sub-devices'.

For a long time the framework was limited to the video_device struct for creating V4L device nodes and video_buf for handling the video buffers (note that this document does not discuss the video_buf framework).

This meant that all drivers had to do the setup of device instances and connecting to sub-devices themselves. Some of this is quite complicated to do right and many drivers never did do it correctly.

There is also a lot of common code that could never be refactored due to the lack of a framework.

So this framework sets up the basic building blocks that all drivers need and this same framework should make it much easier to refactor common code into utility functions shared by all drivers.

A good example to look at as a reference is the v4l2-pci-skeleton.c source that is available in samples/v4l/. It is a skeleton driver for a PCI capture card, and demonstrates how to use the V4L2 driver framework. It can be used as a template for real PCI video capture driver.

Structure of a V4L driver

All drivers have the following structure:

1. A struct for each device instance containing the device state.
2. A way of initializing and commanding sub-devices (if any).
3. Creating V4L2 device nodes (/dev/videoX, /dev/vbiX and /dev/radioX) and keeping track of device-node specific data.
4. Filehandle-specific structs containing per-filehandle data;
5. video buffer handling.

This is a rough schematic of how it all relates:

System Message: WARNING/2 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\[linux-master][Documentation][driver-api][media]v4l2-intro.rst`, line 55)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    device instances
      |
      +-sub-device instances
      |
      \-V4L2 device nodes
          |
          \-filehandle instances
```

Structure of the V4L2 framework

The framework closely resembles the driver structure: it has a v4l2_device struct for the device instance data, a v4l2_subdev struct to refer to sub-device instances, the video_device struct stores V4L2 device node data and the v4l2_fh struct keeps track of filehandle instances.

The V4L2 framework also optionally integrates with the media framework. If a driver sets the struct v4l2_device mdev field, sub-devices and video nodes will automatically appear in the media framework as entities.