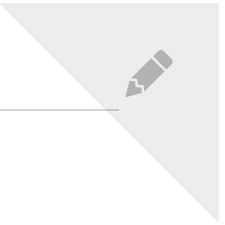
Make camera controls state-aware & streams-aware



Current *state* of V4L2 controls :-)

- Controls for a video device (`/dev/video0`) like a UVC camera
- Controls for a subdev device (`/dev/v4l-subdev0`) like CSI-2 sensor, bridge or ISP
- Can't specify pad or stream, all controls are for the whole device
- Single `struct v4l2_ctrl_handler` for the device
- Same IOCTL for both video and subdev devices:

```
int ioctl(int fd, VIDIOC_S_CTRL, struct v4l2_control *argp)

struct v4l2_control {
    __u32           id;
    __s32           value;
};
```

- Both subdev and device ioctl calls trigger `drivers/media/v4l2-core/v4l2-ctrls-api.c: v4l2_g_ctrl()` :

```
drivers/media/v4l2-core/v4l2-ioctl.c|2363 col 10-21| return v4l2_g_ctrl(vfd->ctrl_handler, p);
drivers/media/v4l2-core/v4l2-subdev.c|669 col 10-21| return v4l2_g_ctrl(vfh->ctrl_handler, arg);
```

  which does a cached lookup through the linked list of controls defined in `struct v4l2_ctrl_handler`
- Video device already supports a per-filehandle control handler, which is used by applications for subscribing to V4L2 events on

a per-filehandle basis
https://lore.kernel.org/linux-media/1307459123-17810-1-git-send-email-hverkuil@xs4all.nl/

- Nothing similar exists for subdev devices

Motivation

Why state-aware?

- For "trying" controls on a subdev along with formats
  - Eg. `HFLIP` and `VFLIP` can affect the allowed bayer patterns
- For atomic updates across the media graph (multi-context ISP support)
  - Also need: `struct video_device_state {}`

Why streams-aware?

- For sensors that transmit multiple pixel streams with different exposure and gain that need to be controlled separately, for eg. A/B mode RGB-Ir sensors like OV2312, OX05B1S

Any other usecases?

Proposal

- Introduce new IOCTL, which matches `VIDIOC_SUBDEV_[GS]_FMT`:

```
ioctl(fd, VIDIOC_SUBDEV_S_CTRL, struct v4l2_subdev_control *argp)

struct v4l2_subdev_control {
    /* control id and value */
    __u32           id;
    __s32           value;
    /* whence -> enum: CONTROL_TRY or CONTROL_ACTIVE */
    __u32           which;
    /* pad and stream */
    __u32           pad;
    __u32           stream;
    /* drivers and applications must zero this array */
    __u32           reserved[x];
};
```

- Store multiple `v4l2_ctrl_handler` inside subdev state, for each pad/stream:

```
struct v4l2_subdev_state {
    /* ... */
    struct v4l2_subdev_pad_config *pads;
    struct v4l2_subdev_krouting routing;
    struct v4l2_subdev_stream_configs stream_configs;
    /* Control configuration for each pad/stream */
    struct v4l2_subdev_ctrl_configs ctrl_configs;
};

struct v4l2_subdev_ctrl_configs {
    u32 num_configs;
    struct v4l2_subdev_ctrl_config *configs;
```

```
};

struct v4l2_subdev_ctrl_config {
    u32 pad;
    u32 stream;
    bool enabled;
    /* Per-stream control handler */
    struct v4l2_ctrl_handler *ctrl_handler;

};
```

Known unknowns

- Should the new controls be per-pad or per-stream?
    - Per-pad could work if it is mandatory to use internal pads for each stream on source devices. But on intermediary subdevs you won't be able to control on a per-stream basis.
- Not all controls make sense for a per-pad basis, for example link frequency.
- Is the new ioctl API (for trying controls) also useful for video devices? Or is that already covered by the per-filehandle control handler?

Unknown unknowns...?