V4L2: Create Better IOCTLs

Hans Verkuil
Cisco Systems Norway

Why?

- Most core V4L2 ioctls got merged in 2002 (design started in 1998) and so are quite old.
- Some ioctls are showing their age and are awkward to use or are out of reserved fields making it difficult to add new features.
- If a new ioctl is added, then the old ioctl should be reimplemented on top of the new one and all drivers should be converted to use the new ioctl(s).
- High level discussion.

VIDIOC_G/S_PARM

- It's really only used for getting/setting the frameinterval.
- Create a VIDIOC_G/S_FRAME_INTERVAL ioctl to just get/set the frameinterval and reimplement the old G/S_PARM on top of that.

struct v4l2_frmivalenum

```
struct v4l2_frmivalenum {
        __u32                   index;          /* Frame format index */
        __u32                   pixel_format;   /* Pixel format */
        __u32                   width;          /* Frame width */
        __u32                   height;         /* Frame height */
        __u32                   type;           /* Frame interval type the device supports. */
        union {                                 /* Frame interval */
                struct v4l2_fract               discrete;
                struct v4l2_frmival_stepwise    stepwise; /* v4l2_fract min, max, step */
        };
        __u32   reserved[2];                    /* Reserved space for future use */
};
```

- 'stepwise' makes little sense for frame intervals. Should we drop support for this? i.e., only support discrete or continuous frame intervals.
- Add 'buf_type' field here and in struct v4l2_frmsizeenum: m2m devices currently cannot choose which queue (capture vs output) to enumerate. If buf_type is 0 (default), then map this to the OUTPUT queue type for m2m devices. For other non-m2m devices it maps to the vb2 queue buffer type (there is only one queue).

ENUMFMT/FRAMESIZES/INTERVALS

- Should we make a new ioctl combining all three that returns all possibilities without having to enumerate everything?
- It is possible to pre-calculate this in the v4l2 core using the existing enum driver callbacks.
- Might be big: vivid has 1710 combinations (an extreme example I admit).
- Is it worth the effort? Are there use-cases where this is a real performance bottleneck?

struct v4l2_buffer (current)

```
struct v4l2_buffer {
        __u32                   index;
        __u32                   type;
        __u32                   bytesused;
        __u32                   flags;
        __u32                   field;
        struct timeval          timestamp;
        struct v4l2_timecode    timecode;
        __u32                   sequence;
        __u32                   memory;
        union {
                __u32           offset;
                unsigned long   userptr;
                struct v4l2_plane *planes;
                __s32           fd;
        } m;
        __u32                   length;
        __u32                   reserved2;
        __u32                   reserved;
};
```

```
struct v4l2_plane {
        __u32                   bytesused;
        __u32                   length;
        union {
                __u32           mem_offset;
                unsigned long   userptr;
                __s32           fd;
        } m;
        __u32                   data_offset;
        __u32                   reserved[11];
};
```

struct v4l2_buffer (suggestion)

```
struct v4l2_ext_buffer {
    __u32                   index;
    __u32                   type;
    __u32                   flags;
    __u32                   field;
    __u64                   timestamp; // Y2038 safe
    __u32                   sequence;
    __u32                   memory;
    struct v4l2_ext_plane   planes[VIDEO_MAX_PLANES];
    __u32                   num_planes;
    __u32                   reserved[27];
};
```

```
struct v4l2_ext_plane {
    __u32       bytesused;
    __u32       length;
    union {
        __u32   mem_offset;
        __u64   userptr;
        __s32   fd;
    } m;
    __u32       data_offset;
    __u32       reserved[11];
};
```

- I want to improve plane description: offset at start of plane, padding at end of plane, 'fd' can refer to one plane or all planes (set to -1 for remaining planes, if allowed by the SoC (exynos4!)).
- Ideas for new fields: u64 cookie (for m2m devices), width/height to simplify handling resolution changes, per-plane 'pixelformat'?
- NV12 vs NV12M: still needed with better buffer description? How to handle backward compat?

struct v4l2_create_buffers

```
struct v4l2_create_buffers {
        __u32                   index;
        __u32                   count;
        __u32                   memory;
        struct v4l2_format      format;
        __u32                   reserved[8];
};
```

- Only format.pix.sizeimage is used (or the corresponding field for other buffer types).
- Originally intention was that drivers would check the whole format (similar to VIDIOC_TRY_FMT), but that never happened (too complex).
- Would like to have something more generic:
  - VIDIOC_CREATE_BUF with just size per plane instead of format (and if 0, use the current format's size like REQBUFS).
  - VIDIOC_DELETE_BUF to delete a specific buffer
  - VIDIOC_DELETE_ALL_BUFS to delete all buffers (or fold into VIDIOC_DELETE_BUF somehow?)

struct v4l2_format & VIDIOC_G/S/TRY_FMT

- v4l2_pix_format vs v4l2_pix_format_mplane: painful
- Running out of fields.
- Improve format descriptions: better able to handle padding/alignment requirements without requiring a new pixel format.
- DRM has u64 format modifiers: add support for those?
- Use ACTIVE and TRY formats, as we do for subdevs?
- Interaction between crop/compose/fmt?