Media controller
Goals, architecture and roadmap

V4L2 Helsinki Summit 2010-06-14

Laurent Pinchart
laurent.pinchart@ideasonboard.com
Embedded SoC camera
- soc_camera
- v4l2_device
- v4l2_subdev

Userspace library
- Format conversion
- Post-processing

Embedded camera
2010-06-14
Media controller - V4L2 Helsinki Summit

- In-kernel functional abstraction layer developed by Hans Verkuil
- Designed for on-board external devices (sensors, tuners, audio codecs, …)
- Reusability, Reusability, Reusability

MT9M001 sensor → PXA270 SoC
MT9M001 sensor → EM28xx USB
SAA7114 decoder → EM28xx USB
SAA7114 decoder → Zoran PCI

V4L2 subdevice
struct v4l2_subdev_ops {
    const struct v4l2_subdev_core_ops *core;
    const struct v4l2_subdev_tuner_ops *tuner;
    const struct v4l2_subdev_audio_ops *audio;
    const struct v4l2_subdev_video_ops *video;
    const struct v4l2_subdev_ir_ops *ir;
    const struct v4l2_subdev_sensor_ops *sensor;
};

- Hardware independent
- Bus type independent
V4L2 device

- Register hardware device
  - i2c_new_device

V4L2 subdevice

- device_driver::probe called

Subdevice initialization

- Retrieve subdevice pointer
  - i2c_get_client_data

- Register subdevice
  - v4l2_device_register_subdev

v4l2_subdev_call(core::s_config)

- Subdevice setup
  - v4l2_i2c_new_subdev_board

V4L2 subdevice registration
SoC Camera

Parallel interface

Host bridge

Sensor

Master clock
Pixel clock
Horizontal sync
Vertical sync
Field
Data[7:0]

I2C

Serial interface (CSI)

Host bridge

Sensor

Master clock
C+
C-
D+
D-

I2C

2010-06-14
V4L2 device

V4L2 call (VIDIOC_S_FMT)

v4l2_subdev_call(video::s_fmt)

Configure the sensor scaler

Success ?

return

Failure ?

Configure the bridge scaler

• How do we decide where to perform scaling ?
  - On the sensor side for higher frame rates ?
  - On the host side for better quality ?
Highly complex devices
- Multiple inputs
- Multiple streams
- Configurable pipeline

Embedded mess
OMAP3430 ISP

- Reconfigurable pipeline
- Parallel processing
- Memory-to-memory paths
- Fine-grain parameters

How do we handle the zillion configuration options through a single video device?

Drawing is © Texas Instrument
Media controller – What and why

**What?**
- Just a device. Similar to v4l2_device, but more abstract.

**Why?**
- To let userspace applications have fine grain control over all the media device parameters.
How?

- Expose the media device topology to userspace as a graph of building blocks called **entities** connected through **pads**.
- Activate/deactivate **links** from userspace.
- Give access to entities **internal parameters** through read/write/ioctl calls.
- Configure image **streaming parameters** at each pad.
struct media_entity
{
    u32 id;
    const char *name;
    u32 type;
    u32 subtype;
    ...
};

- media_entity::type
  - MEDIAENTITYTYPE_NODE
  - MEDIAENTITYTYPE_SUBDEV
struct media_entity
{
    ...
    u8 num_pads;
    struct media_entity_pad *pads;
    ...
};

struct media_entity_pad
{
    u32 type;
    u32 index;
};

- media_entity_pad::type
  - MEDIA_PAD_TYPE_INPUT
  - MEDIA_PAD_TYPE_OUTPUT
struct media_entity
{
    ...
    u32 num_links;
    struct media_entity_link *links;
    ...
};

struct media_entity_link
{
    struct media_entity_pad *source;
    struct media_entity_pad *sink;
    u32 flags;
};

- media_entity_link::flags
  - MEDIA_LINK_FLAG_ACTIVE
  - MEDIA_LINK_FLAG_IMMUTABLE
Media entity

- **Initialize entity**
  - media_entity_init

- **Create links**
  - media_entity_create_link

- **Register entity**
  - media_device_register_entity

---

**Media entity registration**

2010-06-14
int fd;

fd = open("/dev/media0", O_RDWR);
int fd;

fd = open("/dev/media0", O_RDWR);

while (1) {
    struct media_user_entity entity;
    struct media_user_links links;

    ret = ioctl(fd, MEDIA_IOC_ENUM_ENTITIES, &entity);
    if (ret < 0)
        break;

    while (1) {
        ret = ioctl(fd, MEDIA_IOC_ENUM_LINKS, &links);
        if (ret < 0)
            break;
    }
}
Logitech Portable Webcam C905
struct media_user_link link;

link.source.entity = OMAP3_ISP_ENTITY_CCDC;
link.source.index = 2;
link.sink.entity = OMAP3_ISP_ENTITY_PREVIEW;
link.sink.index = 0;
link.flags = 0;

ioctl(fd, MEDIA_IOC_SETUP_LINK, &link);

link.source.entity = OMAP3_ISP_ENTITY_CCDC;
link.source.index = 1;
link.sink.entity = OMAP3_ISP_ENTITY_CCDC_OUT;
link.sink.index = 0;
link.flags = MEDIA_LINK_FLAG_ACTIVE;

ioctl(fd, MEDIA_IOC_SETUP_LINK, &link);
The S_CTRL API is not up to the job.
OMAP3430 ISP - Resizing

- Binning
- Horizontal averaging
- Polyphase filter
OMAP3430 ISP – High quality
The `S_FMT/S_CROP` API is not up to the job.
V4L2 subdev userspace API
struct v4l2_subdev_pad_ops {
    ...
    int (*get_fmt)(struct v4l2_subdev *sd, unsigned int pad,
                   struct v4l2_format *fmt,
                   enum v4l2_subdev_format which);
    int (*set_fmt)(struct v4l2_subdev *sd, unsigned int pad,
                   struct v4l2_format *fmt,
                   enum v4l2_subdev_format which);
    ...
};

struct v4l2_subdev_ops {
    ...
    const struct v4l2_subdev_pad_ops *pad;
};
Is this V4L3?

- No, V4L2 is still alive and well
- Best effort to provide V4L2-only compatibility for existing applications (API and ABI)
- Advanced features will require Media Controller

OMAP3430 ISP

- Default pipeline through /dev/video0?
- Limited set of resolutions, limited set of controls
Retrieve statistics
V4L2 events API

Compute parameters
Host-side software algorithm

Set parameters
VIDIOC_S_CTRL

Image quality
- Image enhancement algorithms
- Hardware-specific acceleration
- Transparent for applications
• http://gitorious.org/omap3camera
• http://git.ideasonboard.org/?p=media-ctl.git