Embedded Alley

Solutions for Intelligent Devices

Video4Linux: What about Output?

Matt Porter

Chief Software Architect

6 April 2009

Introduction

- Video4Linux
  - Introduced in 1997
  - Brought video capture drives under a unified interface
  - Usually considered as a video capture framework
    - Cameras
    - TV tuners
    - Maybe even radio
- Where does streaming video output hardware fit in Linux?
  - Common to multimedia SoCs

Video Output

- What about Video Output Devices???

All Alone Again

V4L Output Devices

- Well documented in the V4L2 specification
  - Be sure you have most recent spec
    - http://v4l2spec.bytesex.org/
- Opposite data flow of a V4l input device (surprise!)
  - Buffers of data in specified pixel format are fed to the V4L device
  - Output device is normally used to send the resultant video stream to an external analog/digital video interface
  - If you have a "special" device, the stock V4L video standards don't always make sense
  - The same is true for enumerating outputs, a special device might just have an internal buffer as a "physical output connector"

V4L Output Devices

- Basics
  - Usual V4L boilerplate required functions
    - Capabilities
    - Output enumeration/get/set
    - Standards enumeration/get/set
    - Formats enumeration/get/set
  - VIDIOC_S_FMT with buffer type V4L_BUF_VIDEO_OUTPUT to define output video source
    - Inserts the video buffer stream directly into the video output signal
  - Optionally use VIDIOC_S_CROP with with buffer type V4L_BUF_VIDEO_OUTPUT to define output cropping rectangle
    - Allows the video buffer stream to be cropped when inserting into the video output signal

Video Output Device Applications

- Maps well to studio and head end video processing equipment
- Stream processing hardware with multiple channels of analog/digital capture/output interfaces
  - Capture NTSC/PAL/SECAM, HDTV, BT656 via V4L input devices
  - Process those streams in user space or via hardware offload
  - Display NTSC/PAL/SECAM, HDTV, BT656 via V4L output devices

Video Output Overlay Devices

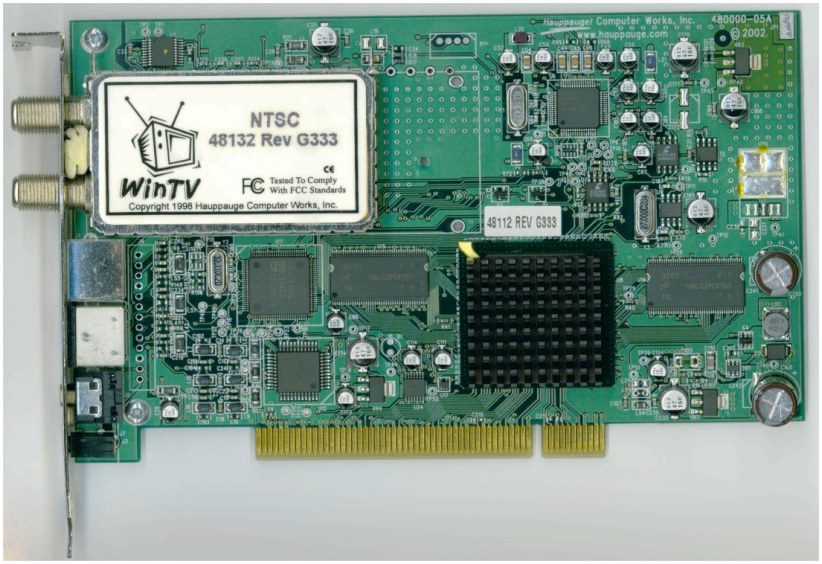
- Used to control video output OSD (On Screen Display) hardware functionality
  - Hardware feature allowing a framebuffer to overlay on top of a video stream
  - Framebuffer and video hardware are typically tightly coupled
- Basics
  - VIDIOC_S_FMT with buffer type V4L_BUF_VIDEO_OUTPUT to define output video source
    - Defines size and pixel format of the video buffers the same as a normal output device.

Video Output Overlay Devices

- VIDIOC_S_FMT with buffer type V4L_BUF_VIDEO_OUTPUT_OVERLAY to define output source cropping/scaling rectangle
  - Allows a subset of the of the output source buffer stream to be selected
  - Alpha blending and chromakey configuration
- VIDIOC_S_CROP with buffer type V4L_BUF_VIDEO_OUTPUT_OVERLAY to define output target cropping/scaling rectangle
  - Selects a target rectangle origin and size to be inserted into the output video stream
- VIDIOC_S_FBUF with buffer type V4L_BUF_VIDEO_OUTPUT_OVERLAY to set overlay configuration
  - Enable/disable FB overlay, alpha blending, and chromakey features

Video Output Overlay Device Applications

- PVR-350
  - The OSD display is what prompted inclusion of output overlay devices into V4L2
  - Supported by the well-known ivtv driver, this is a good reference for implementing a new output overlay driver
    - Driver provides both input/output v4l devices as well as output overlay support for both mpeg and YUV streams



Video Output Overlay Device Applications

- Modern SoCs often support OSD-like functionality
  - This is usually found in a system that supports some video processing that is tightly coupled with the LCD controller and FB engine.
  - Any time the framebuffer allows hardware overlay on a video stream…an output overlay driver is a good match
  - Example: new (unreleased) multimedia SoC…

Pixel Pipeline Hardware

- Pixel pipeline hardware
  - Supports several RGB and YUV formats as source input
  - Output to DRAM buffer that may be used to drive LCD
  - Can crop/scale source input to target buffer
  - Allows hardware-based alpha blending and chromakeying
  - Supports hardware vertical and horizontal flipping
  - Supports hardware rotation in 90 degree increments
  - Supports overlay of frame buffer on the target buffer

Pixel Pipeline Driver

- Perfect match for an output overlay driver
- Most hardware features map 1:1 with V4L APIS
  - Crop/scale h/w maps to the the V4L overlay S_FMT and S_CROP interfaces
  - All pixel formats map to standard V4L formats
  - Flipping controls already exist in V4L
  - Private rotation control is added
  - Handling the FB interaction is the only special part

Pixel Pipeline Driver

- Linux FB driver for pixel pipeline SoC FB Hardware
  - Extended to provide an interface where the V4L pixel pipeline driver can retrieve information on the current var/fix FB settings
  - Allows the V4L driver to limit cropping/scaling of the video stream to the visible area of the FB resolution
    - Because this is the resolution of an attached LCD or NTSC/PAL output
  - VIDIOC_S_FBUF then allows one to enable visibility of the Linux FB contents over a video stream
    - Engaging the overlay may result in no visibility of video or no visibility of FB contents. This depends on use of global/local alpha and chromakey features.
    - It's up to the user to set alpha level appropriately for viewing

Using the Pixel Pipeline in an Application

- How do we use an output overlay driver in application?
  - Unfortunately, there's not a lot of existing support to leverage
  - This results from the history of most V4L drivers being capture type devices
- Libv4l also has mostly support for capture devices
  - But some people are looking at adding output support
- It's however, easy to do basic tests with the simple output overlay API and a command line application
  - Feed RGB/YUV streams to verify the driver
- Wait!, I want to leverage this stuff from standard Linux video frameworks!

Xv

- We can look back to the ivtv driver for a nice example
  - There is an x.org Xv ivtv driver which uses the ivtv output overlay driver to implement Xv support
  - This is nice because the userspace driver wraps around the standard V4L output overlay API and requires no banging directly on h/w in userspace
- An Xv driver allows immediate access to hardware accelerate colorspace conversion for embedded systems based on X11 for the UI
  - Leverages existing Xv output paths in mplayer, gstreamer, etc.

Gstreamer

- Gstreamer has an Xv sink already
- A direct V4L Video Output Overlay sink could be created
  - Would allow direct display of hardware color space conversion accelerated video to a display device without X11 support
  - Gst sink parameters would allow control over standard output overlay features
    - Global alpha
    - Chromakeying
    - Flipping
    - Rotation

DirectFB

- V4L Output Overlay can be supported in the DirectFB framework
- Support exists now for a Davinci driver with support for OSD and hardware blending
  - This can be used as the basis for a generic V4L Output Overlay DirectFB driver

Android

- Yeah, you didn't think we'd get out of here without mentioning Android, right?
- Android's SurfaceFlinger has support for hardware acceleration
  - Overlays
  - Rotation
  - Flipping
  - Hardware blending
- Of course, anybody working with Android knows that this is all **constantly** evolving…

mplayer

- Mplayer has a pretty extensive list of video output modules
- There is even a V4L video output module, but it is specific to one type of hardware with a specific input format
  - This can be abstracted to work with any V4L output overlay driver that comforms to the API
- This will allow for full/scaled/cropped output to a display device without any type of graphics stack
- Work is in progress to implement this generic output overlay module

Conclusion

- Video output overlay devices are often overlooked
  - They are relatively rare compared to capture devices
  - Based on video4linux ML discussions, people are often unaware that they exist.
- The API introduced on behalf of the ivtv is marked experimental but it fits well for many types of hardware
  - The common API ensures that application code will be able to be shared in the future.
  - As support of output overlay hardware increases, many drivers will be able to leverage common code in the various FOSS graphics and video frameworks
- Well known SoC architectures with similar hardware
  - OMAP
  - i.MX

Q&A

- Questions?