Id
Mike Isely <isely@pobox.com>

pvrusb2 driver

Background:

This driver is intended for the "Hauppauge WinTV PVR USB 2.0", which is a USB 2.0 hosted TV Tuner. This driver is a work in progress. Its history started with the reverse-engineering effort by Björn Danielsson <pvrusb2@dax.nu> whose web page can be found here:

http://pvrusb2.dax.nu/

From there Aurelien Alleaume <slts@free.fr> began an effort to create a video4linux compatible driver. I began with Aurelien's last known snapshot and evolved the driver to the state it is in here.

More information on this driver can be found at:

http://www.isely.net/pvrusb2.html

This driver has a strong separation of layers. They are very roughly:

1a. Low level wire-protocol implementation with the device.

1b. I2C adaptor implementation and corresponding I2C client drivers implemented elsewhere in V4L.

1c. High level hardware driver implementation which coordinates all activities that ensure correct operation of the device.

2. A "context" layer which manages instancing of driver, setup, tear-down, arbitration, and interaction with high level interfaces appropriately as devices are hotplugged in the system.

3. High level interfaces which glue the driver to various published Linux APIs (V4L, sysfs, maybe DVB in the future).

The most important shearing layer is between the top 2 layers. A lot of work went into the driver to ensure that any kind of conceivable API can be laid on top of the core driver. (Yes, the driver internally leverages V4L to do its work but that really has nothing to do with the API published by the driver to the outside world.) The architecture allows for different APIs to simultaneously access the driver. I have a strong sense of fairness about APIs and also feel that it is a good design principle to keep implementation and interface isolated from each other. Thus while right now the V4L high level interface is the most complete, the sysfs high level interface will work equally well for similar functions, and there's no reason I see right now why it shouldn't be

possible to produce a DVB high level interface that can sit right alongside V4L.

NOTE: Complete documentation on the pvrusb2 driver is contained in the html files within the doc directory; these are exactly the same as what is on the web site at the time. Browse those files (especially the FAQ) before asking questions.

Building

To build these modules essentially amounts to just running "Make", but you need the kernel source tree nearby and you will likely also want to set a few controlling environment variables first in order to link things up with that source tree. Please see the Makefile here for comments that explain how to do that.

Source file list / functional overview:

(Note: The term "module" used below generally refers to loosely defined functional units within the pvrusb2 driver and bears no relation to the Linux kernel's concept of a loadable module.)

pvrusb2-audio.[ch] - This is glue logic that resides between this driver and the msp3400.ko I2C client driver (which is found elsewhere in V4L).

pvrusb2-context.[ch] - This module implements the context for an instance of the driver. Everything else eventually ties back to or is otherwise instanced within the data structures implemented here. Hotplugging is ultimately coordinated here. All high level interfaces tie into the driver through this module. This module helps arbitrate each interface's access to the actual driver core, and is designed to allow concurrent access through multiple instances of multiple interfaces (thus you can for example change the tuner's frequency through sysfs while simultaneously streaming video through V4L out to an instance of mplayer).

pvrusb2-debug.h - This header defines a printk() wrapper and a mask of debugging bit definitions for the various kinds of debug messages that can be enabled within the driver.

pvrusb2-debugifc.[ch] - This module implements a crude command line oriented debug interface into the driver. Aside from being part of the process for implementing manual firmware extraction (see the pvrusb2 web site mentioned earlier), probably I'm the only one who has ever used this. It is mainly a debugging aid.

pvrusb2-eeprom.[ch] - This is glue logic that resides between this driver the tveeprom.ko module, which is itself implemented elsewhere in V4L.

pvrusb2-encoder.[ch] - This module implements all protocol needed to interact with the Conexant mpeg2 encoder chip within the pvrusb2 device. It is a crude echo of corresponding logic in ivtv,

however the design goals (strict isolation) and physical layer (proxy through USB instead of PCI) are enough different that this implementation had to be completely different.

pvrusb2-hdw-internal.h - This header defines the core data structure in the driver used to track ALL internal state related to control of the hardware. Nobody outside of the core hardware-handling modules should have any business using this header. All external access to the driver should be through one of the high level interfaces (e.g. V4L, sysfs, etc), and in fact even those high level interfaces are restricted to the API defined in pvrusb2-hdw.h and NOT this header.

pvrusb2-hdw.h - This header defines the full internal API for controlling the hardware. High level interfaces (e.g. V4L, sysfs) will work through here.

pvrusb2-hdw.c - This module implements all the various bits of logic that handle overall control of a specific pvrusb2 device. (Policy, instantiation, and arbitration of pvrusb2 devices fall within the jurisdiction of pvrusb-context not here).

pvrusb2-i2c-chips-*.c - These modules implement the glue logic to tie together and configure various I2C modules as they attach to the I2C bus. There are two versions of this file. The "v4l2" version is intended to be used in-tree alongside V4L, where we implement just the logic that makes sense for a pure V4L environment. The "all" version is intended for use outside of V4L, where we might encounter other possibly "challenging" modules from ivtv or older kernel snapshots (or even the support modules in the standalone snapshot).

pvrusb2-i2c-cmd-v4l1.[ch] - This module implements generic V4L1 compatible commands to the I2C modules. It is here where state changes inside the pvrusb2 driver are translated into V4L1 commands that are in turn send to the various I2C modules.

pvrusb2-i2c-cmd-v4l2.[ch] - This module implements generic V4L2 compatible commands to the I2C modules. It is here where state changes inside the pvrusb2 driver are translated into V4L2 commands that are in turn send to the various I2C modules.

pvrusb2-i2c-core.[ch] - This module provides an implementation of a kernel-friendly I2C adaptor driver, through which other external I2C client drivers (e.g. msp3400, tuner, lirc) may connect and operate corresponding chips within the pvrusb2 device. It is through here that other V4L modules can reach into this driver to operate specific pieces (and those modules are in turn driven by glue logic which is coordinated by pvrusb2-hdw, doled out by pvrusb2-context, and then ultimately made available to users through one of the high level interfaces).

pvrusb2-io.[ch] - This module implements a very low level ring of transfer buffers, required in order to stream data from the device. This module is *very* low level. It only operates the buffers and makes no attempt to define any policy or mechanism for

how such buffers might be used.

pvrusb2-ioread.[ch] - This module layers on top of pvrusb2-io.[ch] to provide a streaming API usable by a read() system call style of I/O. Right now this is the only layer on top of pvrusb2-io.[ch], however the underlying architecture here was intended to allow for other styles of I/O to be implemented with additonal modules, like mmap()'ed buffers or something even more exotic.

pvrusb2-main.c - This is the top level of the driver. Module level and USB core entry points are here. This is our "main".

pvrusb2-sysfs.[ch] - This is the high level interface which ties the pvrusb2 driver into sysfs. Through this interface you can do everything with the driver except actually stream data.

pvrusb2-tuner.[ch] - This is glue logic that resides between this driver and the tuner.ko I2C client driver (which is found elsewhere in V4L).

pvrusb2-util.h - This header defines some common macros used throughout the driver. These macros are not really specific to the driver, but they had to go somewhere.

pvrusb2-v4l2.[ch] - This is the high level interface which ties the pvrusb2 driver into video4linux. It is through here that V4L applications can open and operate the driver in the usual V4L ways. Note that **ALL** V4L functionality is published only through here and nowhere else.

pvrusb2-video-*.[ch] - This is glue logic that resides between this driver and the saa711x.ko I2C client driver (which is found elsewhere in V4L). Note that saa711x.ko used to be known as saa7115.ko in ivtv. There are two versions of this; one is selected depending on the particular saa711[5x].ko that is found.

pvrusb2.h - This header contains compile time tunable parameters (and at the moment the driver has very little that needs to be tuned).

-Mike Isely
isely@pobox.com