Frequently Asked Questions:
==========================
subject: unified zoran driver (zr360x7, zoran, buz, dc10(+), dc30(+), lml33)
website: http://mjpeg.sourceforge.net/driver-zoran/

==========================

1. What cards are supported

Iomega Buz, Linux Media Labs LML33/LML33R10, Pinnacle/Miro
DC10/DC10+/DC30/DC30+ and related boards (available under various names).

Iomega Buz:
* Zoran zr36067 PCI controller
* Zoran zr36060 MJPEG codec
* Philips saa7111 TV decoder
* Philips saa7185 TV encoder
Drivers to use: videodev, i2c-core, i2c-algo-bit,
videocodec, saa7111, saa7185, zr36060, zr36067
Inputs/outputs: Composite and S-video
Norms: PAL, SECAM (720x576 @ 25 fps), NTSC (720x480 @ 29.97 fps)
Card number: 7

AverMedia 6 Eyes AVS6EYES:
* Zoran zr36067 PCI controller
* Zoran zr36060 MJPEG codec
* Samsung ks0127 TV decoder
* Conexant bt866 TV encoder
Drivers to use: videodev, i2c-core, i2c-algo-bit,
videocodec, ks0127, bt866, zr36060, zr36067
Inputs/outputs: Six physical inputs. 1-6 are composite,
1-2, 3-4, 5-6 doubles as S-video,
1-3 triples as component.
One composite output.
Norms: PAL, SECAM (720x576 @ 25 fps), NTSC (720x480 @ 29.97 fps)
Card number: 8
Not autodetected, card=8 is necessary.

Linux Media Labs LML33:
* Zoran zr36067 PCI controller
* Zoran zr36060 MJPEG codec
* Brooktree bt819 TV decoder
* Brooktree bt856 TV encoder
Drivers to use: videodev, i2c-core, i2c-algo-bit,

videocodec, bt819, bt856, zr36060, zr36067
Inputs/outputs: Composite and S-video
Norms: PAL (720x576 @ 25 fps), NTSC (720x480 @ 29.97 fps)
Card number: 5

Linux Media Labs LML33R10:
* Zoran zr36067 PCI controller
* Zoran zr36060 MJPEG codec
* Philips saa7114 TV decoder
* Analog Devices adv7170 TV encoder
Drivers to use: videodev, i2c-core, i2c-algo-bit,
videocodec, saa7114, adv7170, zr36060, zr36067
Inputs/outputs: Composite and S-video
Norms: PAL (720x576 @ 25 fps), NTSC (720x480 @ 29.97 fps)
Card number: 6

Pinnacle/Miro DC10(new):
* Zoran zr36057 PCI controller
* Zoran zr36060 MJPEG codec
* Philips saa7110a TV decoder
* Analog Devices adv7176 TV encoder
Drivers to use: videodev, i2c-core, i2c-algo-bit,
videocodec, saa7110, adv7175, zr36060, zr36067
Inputs/outputs: Composite, S-video and Internal
Norms: PAL, SECAM (768x576 @ 25 fps), NTSC (640x480 @ 29.97 fps)
Card number: 1

Pinnacle/Miro DC10+:
* Zoran zr36067 PCI controller
* Zoran zr36060 MJPEG codec
* Philips saa7110a TV decoder
* Analog Devices adv7176 TV encoder
Drivers to use: videodev, i2c-core, i2c-algo-bit,
videocodec, sa7110, adv7175, zr36060, zr36067
Inputs/outputs: Composite, S-video and Internal
Norms: PAL, SECAM (768x576 @ 25 fps), NTSC (640x480 @ 29.97 fps)
Card number: 2

Pinnacle/Miro DC10(old): *
* Zoran zr36057 PCI controller
* Zoran zr36050 MJPEG codec
* Zoran zr36016 Video Front End or Fuji md0211 Video Front End (clone?)
* Micronas vpx3220a TV decoder
* mse3000 TV encoder or Analog Devices adv7176 TV encoder *
Drivers to use: videodev, i2c-core, i2c-algo-bit,
videocodec, vpx3220, mse3000/adv7175, zr36050, zr36016, zr36067
Inputs/outputs: Composite, S-video and Internal
Norms: PAL, SECAM (768x576 @ 25 fps), NTSC (640x480 @ 29.97 fps)
Card number: 0

Pinnacle/Miro DC30: *
* Zoran zr36057 PCI controller
* Zoran zr36050 MJPEG codec
* Zoran zr36016 Video Front End
* Micronas vpx3225d/vpx3220a/vpx3216b TV decoder
* Analog Devices adv7176 TV encoder

Drivers to use: videodev, i2c-core, i2c-algo-bit,
videocodec, vpx3220/vpx3224, adv7175, zr36050, zr36016, zr36067
Inputs/outputs: Composite, S-video and Internal
Norms: PAL, SECAM (768x576 @ 25 fps), NTSC (640x480 @ 29.97 fps)
Card number: 3

Pinnacle/Miro DC30+: *
* Zoran zr36067 PCI controller
* Zoran zr36050 MJPEG codec
* Zoran zr36016 Video Front End
* Micronas vpx3225d/vpx3220a/vpx3216b TV decoder
* Analog Devices adv7176 TV encoder
Drivers to use: videodev, i2c-core, i2c-algo-bit,
videocodec, vpx3220/vpx3224, adv7175, zr36050, zr36015, zr36067
Inputs/outputs: Composite, S-video and Internal
Norms: PAL, SECAM (768x576 @ 25 fps), NTSC (640x480 @ 29.97 fps)
Card number: 4

Note: No module for the mse3000 is available yet
Note: No module for the vpx3224 is available yet
Note: use encoder=X or decoder=X for non-default i2c chips (see i2c-id.h)

==========

1.1 What the TV decoder can do an what not

The best know TV standards are NTSC/PAL/SECAM. but for decoding a frame that information is not enough. There are several formats of the TV standards. And not every TV decoder is able to handle every format. Also the every combination is supported by the driver. There are currently 11 different tv broadcast formats all aver the world.

The CCIR defines parameters needed for broadcasting the signal.
The CCIR has defined different standards: A,B,D,E,F,G,D,H,I,K,K1,L,M,N,...
The CCIR says not much about the colorsystem used !!!
And talking about a colorsystem says not to much about how it is broadcast.

The CCIR standards A,E,F are not used any more.

When you speak about NTSC, you usually mean the standard: CCIR - M using the NTSC colorsystem which is used in the USA, Japan, Mexico, Canada and a few others.

When you talk about PAL, you usually mean: CCIR - B/G using the PAL colorsystem which is used in many Countries.

When you talk about SECAM, you mean: CCIR - L using the SECAM Colorsystem which is used in France, and a few others.

There the other version of SECAM, CCIR - D/K is used in Bulgaria, China, Slovakai, Hungary, Korea (Rep.), Poland, Rumania and a others.

The CCIR - H uses the PAL colorsystem (sometimes SECAM) and is used in Egypt, Libya, Sri Lanka, Syrain Arab. Rep.

The CCIR - I uses the PAL colorsystem, and is used in Great Britain, Hong Kong,

Ireland, Nigeria, South Africa.

The CCIR - N uses the PAL colorsystem and PAL frame size but the NTSC framerate, and is used in Argentinia, Uruguay, an a few others

We do not talk about how the audio is broadcast !

A rather good sites about the TV standards are:
http://www.sony.jp/ServiceArea/Voltage_map/
http://info.electronicwerkstatt.de/bereiche/fernsehtechnik/frequenzen_und_normen/Fernsehnormen/
and http://www.cabl.com/restaurant/channel.html

Other weird things around: NTSC 4.43 is a modificated NTSC, which is mainly used in PAL VCR's that are able to play back NTSC. PAL 60 seems to be the same as NTSC 4.43 . The Datasheets also talk about NTSC 44, It seems as if it would be the same as NTSC 4.43.
NTSC Combs seems to be a decoder mode where the decoder uses a comb filter to split coma and luma instead of a Delay line.

But I did not defiantly find out what NTSC Comb is.

Philips saa7111 TV decoder
was introduced in 1997, is used in the BUZ and
can handle: PAL B/G/H/I, PAL N, PAL M, NTSC M, NTSC N, NTSC 4.43 and SECAM

Philips saa7110a TV decoder
was introduced in 1995, is used in the Pinnacle/Miro DC10(new), DC10+ and
can handle: PAL B/G, NTSC M and SECAM

Philips saa7114 TV decoder
was introduced in 2000, is used in the LML33R10 and
can handle: PAL B/G/D/H/I/N, PAL N, PAL M, NTSC M, NTSC 4.43 and SECAM

Brooktree bt819 TV decoder
was introduced in 1996, and is used in the LML33 and
can handle: PAL B/D/G/H/I, NTSC M

Micronas vpx3220a TV decoder
was introduced in 1996, is used in the DC30 and DC30+ and
can handle: PAL B/G/H/I, PAL N, PAL M, NTSC M, NTSC 44, PAL 60, SECAM,NTSC Comb

Samsung ks0127 TV decoder
is used in the AVS6EYES card and
can handle: NTSC-M/N/44, PAL-M/N/B/G/H/I/D/K/L and SECAM

==========

1.2 What the TV encoder can do an what not

The TV encoder are doing the "same" as the decoder, but in the oder direction.
You feed them digital data and the generate a Composite or SVHS signal.
For information about the colorsystems and TV norm take a look in the
TV decoder section.

Philips saa7185 TV Encoder

was introduced in 1996, is used in the BUZ
can generate: PAL B/G, NTSC M

Brooktree bt856 TV Encoder
was introduced in 1994, is used in the LML33
can generate: PAL B/D/G/H/I/N, PAL M, NTSC M, PAL-N (Argentina)

Analog Devices adv7170 TV Encoder
was introduced in 2000, is used in the LML300R10
can generate: PAL B/D/G/H/I/N, PAL M, NTSC M, PAL 60

Analog Devices adv7175 TV Encoder
was introduced in 1996, is used in the DC10, DC10+, DC10 old, DC30, DC30+
can generate: PAL B/D/G/H/I/N, PAL M, NTSC M

ITT mse3000 TV encoder
was introduced in 1991, is used in the DC10 old
can generate: PAL , NTSC , SECAM

Conexant bt866 TV encoder
is used in AVS6EYES, and
can generate: NTSC/PAL, PAL-M, PAL-N

The adv717x, should be able to produce PAL N. But you find nothing PAL N specific in the registers. Seem that you have to reuse a other standard to generate PAL N, maybe it would work if you use the PAL M settings.

==========

2. How do I get this damn thing to work

Load zr36067.o. If it can't autodetect your card, use the card=X insmod option with X being the card number as given in the previous section. To have more than one card, use card=X1[,X2[,X3,[X4[..]]]]

To automate this, add the following to your /etc/modprobe.conf:

options zr36067 card=X1[,X2[,X3[,X4[..]]]]
alias char-major-81-0 zr36067

One thing to keep in mind is that this doesn't load zr36067.o itself yet. It just automates loading. If you start using xawtv, the device won't load on some systems, since you're trying to load modules as a user, which is not allowed ("permission denied"). A quick workaround is to add 'Load "v4l"' to XF86Config-4 when you use X by default, or to run 'v4l-conf -c <device>' in one of your startup scripts (normally rc.local) if you don't use X. Both make sure that the modules are loaded on startup, under the root account.

==========

3. What mainboard should I use (or why doesn't my card work)

<insert lousy disclaimer here>. In short: good=SiS/Intel, bad=VIA.

Experience tells us that people with a Buz, on average, have more problems than users with a DC10+/LML33. Also, it tells us that people owning a VIA-

based mainboard (ktXXX, MVP3) have more problems than users with a mainboard based on a different chipset. Here's some notes from Andrew Stevens:
--
Here's my experience of using LML33 and Buz on various motherboards:

VIA MVP3
        Forget it. Pointless. Doesn't work.
Intel 430FX (Pentium 200)
        LML33 perfect, Buz tolerable (3 or 4 frames dropped per movie)
Intel 440BX (early stepping)
        LML33 tolerable. Buz starting to get annoying (6-10 frames/hour)
Intel 440BX (late stepping)
        Buz tolerable, LML3 almost perfect (occasional single frame drops)
SiS735
        LML33 perfect, Buz tolerable.
VIA KT133(*)
        LML33 starting to get annoying, Buz poor enough that I have up.

Both 440BX boards were dual CPU versions.
--
Bernhard Praschinger later added:
--
AMD 751
        Buz perfect-tolerable
AMD 760
        Buz perfect-tolerable
--
In general, people on the user mailinglist won't give you much of a chance if you have a VIA-based motherboard. They may be cheap, but sometimes, you'd rather want to spend some more money on better boards. In general, VIA mainboard's IDE/PCI performance will also suck badly compared to others. You'll noticed the DC10+/DC30+ aren't mentioned anywhere in the overview. Basically, you can assume that if the Buz works, the LML33 will work too. If the LML33 works, the DC10+/DC30+ will work too. They're most tolerant to different mainboard chipsets from all of the supported cards.

If you experience timeouts during capture, buy a better mainboard or lower the quality/buffersize during capture (see 'Concerning buffer sizes, quality, output size etc.'). If it hangs, there's little we can do as of now. Check your IRQs and make sure the card has its own interrupts.

==========

4. Programming interface

This driver conforms to video4linux and video4linux2, both can be used to use the driver. Since video4linux didn't provide adequate calls to fully use the cards' features, we've introduced several programming extensions, which are currently officially accepted in the 2.4.x branch of the kernel. These extensions are known as the v4l/mjpeg extensions. See zoran.h for details (structs/ioctls).

Information - video4linux:
http://roadrunner.swansea.linux.org.uk/v4lapi.shtml
Documentation/video4linux/API.html
/usr/include/linux/videodev.h

Information - video4linux/mjpeg extensions:
./zoran.h
(also see below)

Information - video4linux2:
http://linuxtv.org
http://v4l2spec.bytesex.org/
/usr/include/linux/videodev2.h

More information on the video4linux/mjpeg extensions, by Serguei Miridonovi and Rainer Johanni:
--
The ioctls for that interface are as follows:

BUZIOC_G_PARAMS
BUZIOC_S_PARAMS

Get and set the parameters of the buz. The user should always do a BUZIOC_G_PARAMS (with a struct buz_params) to obtain the default settings, change what he likes and then make a BUZIOC_S_PARAMS call.

BUZIOC_REQBUFS

Before being able to capture/playback, the user has to request the buffers he is wanting to use. Fill the structure zoran_requestbuffers with the size (recommended: 256*1024) and the number (recommended 32 up to 256). There are no such restrictions as for the Video for Linux buffers, you should LEAVE SUFFICIENT MEMORY for your system however, else strange things will happen ....
On return, the zoran_requestbuffers structure contains number and size of the actually allocated buffers.
You should use these numbers for doing a mmap of the buffers into the user space.
The BUZIOC_REQBUFS ioctl also makes it happen, that the next mmap maps the MJPEG buffer instead of the V4L buffers.

BUZIOC_QBUF_CAPT
BUZIOC_QBUF_PLAY

Queue a buffer for capture or playback. The first call also starts streaming capture. When streaming capture is going on, you may only queue further buffers or issue syncs until streaming capture is switched off again with a argument of -1 to a BUZIOC_QBUF_CAPT/BUZIOC_QBUF_PLAY ioctl.

BUZIOC_SYNC

Issue this ioctl when all buffers are queued. This ioctl will block until the first buffer becomes free for saving its data to disk (after BUZIOC_QBUF_CAPT) or for reuse (after BUZIOC_QBUF_PLAY).

BUZIOC_G_STATUS

Get the status of the input lines (video source connected/norm).

For programming example, please, look at lavrec.c and lavplay.c code in lavtools-1.2p2 package (URL: http://www.cicese.mx/~mirsev/DC10plus/) and the 'examples' directory in the original Buz driver distribution.

Additional notes for software developers:

   The driver returns maxwidth and maxheight parameters according to the current TV standard (norm). Therefore, the software which communicates with the driver and "asks" for these parameters should first set the correct norm. Well, it seems logically correct: TV standard is "more constant" for current country than geometry settings of a variety of TV capture cards which may work in ITU or square pixel format.
--
Please note that lavplay/lavrec are also included in the MJPEG-tools (http://mjpeg.sf.net/).

==========

5. Applications

Applications known to work with this driver:

TV viewing:
* xawtv
* kwintv
* probably any TV application that supports video4linux or video4linux2.

MJPEG capture/playback:
* mjpegtools/lavtools (or Linux Video Studio)
* gstreamer
* mplayer

General raw capture:
* xawtv
* gstreamer
* probably any application that supports video4linux or video4linux2

Video editing:
* Cinelerra
* MainActor
* mjpegtools (or Linux Video Studio)

==========

6. Concerning buffer sizes, quality, output size etc.

The zr36060 can do 1:2 JPEG compression. This is really the theoretical maximum that the chipset can reach. The driver can, however, limit compression to a maximum (size) of 1:4. The reason for this is that some cards (e.g. Buz) can't handle 1:2 compression without stopping capture after only a few minutes. With 1:4, it'll mostly work. If you have a Buz, use 'low_bitrate=1' to go into 1:4 max. compression mode.

100% JPEG quality is thus 1:2 compression in practice. So for a full PAL frame (size 720x576). The JPEG fields are stored in YUY2 format, so the size of the

fields are 720x288x16/2 bits/field (2 fields/frame) = 207360 bytes/field x 2 = 414720 bytes/frame (add some more bytes for headers and DHT (huffman)/DQT (quantization) tables, and you'll get to something like 512kB per frame for 1:2 compression. For 1:4 compression, you'd have frames of half this size.

Some additional explanation by Martin Samuelsson, which also explains the importance of buffer sizes:
--
> Hmm, I do not think it is really that way. With the current (downloaded
> at 18:00 Monday) driver I get that output sizes for 10 sec:
> -q 50 -b 128 : 24.283.332 Bytes
> -q 50 -b 256 : 48.442.368
> -q 25 -b 128 : 24.655.992
> -q 25 -b 256 : 25.859.820

I woke up, and can't go to sleep again. I'll kill some time explaining why this doesn't look strange to me.

Let's do some math using a width of 704 pixels. I'm not sure whether the Buz actually use that number or not, but that's not too important right now.

704x288 pixels, one field, is 202752 pixels. Divided by 64 pixels per block; 3168 blocks per field. Each pixel consist of two bytes; 128 bytes per block; 1024 bits per block. 100% in the new driver mean 1:2 compression; the maximum output becomes 512 bits per block. Actually 510, but 512 is simpler to use for calculations.

Let's say that we specify d1q50. We thus want 256 bits per block; times 3168 becomes 811008 bits; 101376 bytes per field. We're talking raw bits and bytes here, so we don't need to do any fancy corrections for bits-per-pixel or such things. 101376 bytes per field.

d1 video contains two fields per frame. Those sum up to 202752 bytes per frame, and one of those frames goes into each buffer.

But wait a second! -b128 gives 128kB buffers! It's not possible to cram 202752 bytes of JPEG data into 128kB!

This is what the driver notice and automatically compensate for in your examples. Let's do some math using this information:

128kB is 131072 bytes. In this buffer, we want to store two fields, which leaves 65536 bytes for each field. Using 3168 blocks per field, we get 20.68686868... available bytes per block; 165 bits. We can't allow the request for 256 bits per block when there's only 165 bits available! The -q50 option is silently overridden, and the -b128 option takes precedence, leaving us with the equivalence of -q32.

This gives us a data rate of 165 bits per block, which, times 3168, sums up to 65340 bytes per field, out of the allowed 65536. The current driver has another level of rate limiting; it won't accept -q values that fill more than 6/8 of the specified buffers. (I'm not sure why. "Playing it safe" seem to be a safe bet. Personally, I think I would have lowered requested-bits-per-block by one, or something like that.) We can't use 165 bits per block, but have to lower it again, to 6/8 of the available buffer space: We end up with 124 bits per block, the equivalence of -q24. With 128kB buffers, you can't use greater

than -q24 at -d1. (And PAL, and 704 pixels width...)

The third example is limited to -q24 through the same process. The second example, using very similar calculations, is limited to -q48. The only example that actually grab at the specified -q value is the last one, which is clearly visible, looking at the file size.
--

Conclusion: the quality of the resulting movie depends on buffer size, quality, whether or not you use 'low_bitrate=1' as insmod option for the zr36060.c module to do 1:4 instead of 1:2 compression, etc.

If you experience timeouts, lowering the quality/buffersize or using 'low_bitrate=1 as insmod option for zr36060.o might actually help, as is proven by the Buz.

==========

7. It hangs/crashes/fails/whatevers! Help!

Make sure that the card has its own interrupts (see /proc/interrupts), check the output of dmesg at high verbosity (load zr36067.o with debug=2, load all other modules with debug=1). Check that your mainboard is favorable (see question 2) and if not, test the card in another computer. Also see the notes given in question 3 and try lowering quality/buffersize/capturesize if recording fails after a period of time.

If all this doesn't help, give a clear description of the problem including detailed hardware information (memory+brand, mainboard+chipset+brand, which MJPEG card, processor, other PCI cards that might be of interest), give the system PnP information (/proc/interrupts, /proc/dma, /proc/devices), and give the kernel version, driver version, glibc version, gcc version and any other information that might possibly be of interest. Also provide the dmesg output at high verbosity. See 'Contacting' on how to contact the developers.

==========

8. Maintainers/Contacting

The driver is currently maintained by Laurent Pinchart and Ronald Bultje (<laurent.pinchart@skynet.be> and <rbultje@ronald.bitfreak.net>). For bug reports or questions, please contact the mailinglist instead of the developers individually. For user questions (i.e. bug reports or how-to questions), send an email to <mjpeg-users@lists.sf.net>, for developers (i.e. if you want to help programming), send an email to <mjpeg-developer@lists.sf.net>. See http://www.sf.net/projects/mjpeg/ for subscription information.

For bug reports, be sure to include all the information as described in the section 'It hangs/crashes/fails/whatevers! Help!'. Please make sure you're using the latest version (http://mjpeg.sf.net/driver-zoran/).

Previous maintainers/developers of this driver include Serguei Miridonov <mirsev@cicese.mx>, Wolfgang Scherr <scherr@net4you.net>, Dave Perks <dperks@ibm.net> and Rainer Johanni <Rainer@Johanni.de>.

==========

9. License

This driver is distributed under the terms of the General Public License.

> This program is free software; you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation; either version 2 of the License, or
> (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program; if not, write to the Free Software
> Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

See http://www.gnu.org/ for more information.